Politechnika
Śląska

PRACA MAGISTERSKA

Analiza algorytmów proceduralnego generowania poziomów w grach typu roguelike.

Bartłomiej GORDON
Nr albumu: 295650

Kierunek: Informatyka
Specjalność: Interaktywna Grafika Trójwymiarowa

PROWADZĄCY PRACĘ
dr Ewa Lach
KATEDRA Grafiki Wizji Komputerowej i Systemów Cyfrowych
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2025

Tytuł pracy

Analiza algorytmów proceduralnego generowania poziomów w grach typu roguelike.

Streszczenie

Celem pracy jest ocena i porównanie algorytmów proceduralnego generowania poziomów w grach z gatunku roguelike. Ewaluacja wybranych podejść została wykonana z użyciem ujednoliconego zestawu metryk strukturalnych i grywalnościowych, w tym metryk opartych na symulacji gracza. Zaprojektowano środowisko badawcze i procedury eksperymentalne zapewniające powtarzalność oraz skalowalność badań, umożliwiające analizę zależności między parametrami wejściowymi algorytmów a właściwościami wytworzonych poziomów. Uzyskane wyniki pozwalają na ocenę wpływu algorytmów i ich parametrów na wygenerowane poziomy. Praca przedstawia również ograniczenia i wnioski praktyczne dla projektantów gier.

Słowa kluczowe

proceduralne generowanie treści, roguelike, ewaluacja i porównanie algorytmów, symulacja gracza

Thesis title

Analysis of algorithms for procedural level generation in roguelike games.

Abstract

The aim of this study is to evaluate and compare algorithms for procedural content generation in roguelike games. The selected approaches were assessed using a unified set of structural and playability metrics, including metrics based on player simulation. A research environment and experimental procedures were designed to ensure reproducibility and scalability, enabling analysis of the relationships between algorithm input parameters and the properties of the generated levels. The obtained results allow for an assessment of the impact of the algorithms and their parameter settings on the generated levels. The study also discusses limitations and offers practical recommendations for game designers.

Keywords

procedural content generation, roguelike, evaluation and comparison of algorithms, player simulation

Spis treści

Rozdział 1

Wstęp

Współczesne gry komputerowe coraz częściej wykorzystują techniki automatycznego tworzenia zawartości, znane jako proceduralne generowanie treści, PCG (ang. *procedural content generation*). Jednym z kluczowych zastosowań PCG jest generowanie poziomów, czyli automatyczne projektowanie układów przestrzennych, zadań czy wyzwań napotykanych przez gracza. Szczególnie istotne jest to w grach typu roguelike, zaliczanych do gatunku gier komputerowych, w których losowość oraz zróżnicowanie generowanych poziomów odgrywaja kluczową rolę, determinując zarówno wysoki poziom regrywalności, jak i możliwość dostosowania wyzwań do kompetencji gracza. Pomimo rosnącej popularności tych rozwiązań, wciąż brakuje jednoznacznych kryteriów oceny jakości generowanych poziomów oraz porównań skuteczności różnych algorytmów. Problem ten nabiera znaczenia zarówno z perspektywy projektantów gier, jak i badaczy zajmujących się automatyzacją procesu tworzenia treści.

1.1 Cel pracy

Celem pracy jest ocena i porównanie wybranych algorytmów proceduralnego generowania poziomów w postaci lochów, labiryntów i jaskiń. Każdy algorytm reprezentuje odmienną strategię twórczą i operuje na różnym zestawie parametrów wejściowych. Oceny dokonano z użyciem wspólnego zestawu metryk topologicznych opartych na analizie układu poziomu oraz metryk grywalnościowych wyznaczanych poprzez symulację gracza. Trzon badań stanowi eksperyment obliczeniowy obejmujący generowanie dużych prób poziomów w całej przestrzeni parametrów każdego algorytmu. Na potrzeby pracy przygotowano środowisko badawcze, które automatycznie tworzy poziomy, oblicza metryki, uruchamia symulacje i agreguje wyniki.

1.2 Wkład autora

Autor przeprowadził badania ilościowe porównujące algorytmy generowania poziomów. Zaprojektował spójny zestaw metryk i kryteriów oceny, umożliwiający bezpośrednie zestawianie wyników, oraz dobrał algorytmy reprezentujące różne podejścia. Przygotował środowisko badawcze z symulacją wirtualnego gracza i ujednolicił implementacje analizowanych algorytmów. Zrealizował szeroką serię eksperymentów, opracował wyniki z oszacowaniem niepewności, a następnie je przeanalizował i zinterpretował.

1.3 Organizacja pracy

Praca składa się z pięciu rozdziałów. W rozdziale pierwszym przedstawiono kontekst, motywację badań i wkład pracy, a także opisano organizację dokumentu. Rozdział drugi zawiera przegląd algorytmów proceduralnego generowania poziomów w grach z gatunku roguelike i ich oceny (strukturalnych i grywalnościowych). W rozdziale trzecim zaprezentowano środowisko badawcze, wybrane algorytmy i ich parametryzację, architekturę platformy eksperymentalnej, zestaw metryk oraz procedury eksperymentalne zapewniające powtarzalność i skalowalność analiz. Rozdział czwarty przedstawia wyniki eksperymentów wraz z ich analizą, porównanie badanych metod, profilowanie ich zachowania, analizę wrażliwości na parametry wejściowe oraz współzależności metryk. W rozdziale piątym podsumowano pracę.

Dodatkowo do pracy załączono kod źródłowy platformy badawczej wraz ze wszystkimi wynikami badań.

Rozdział 2

Analiza tematu

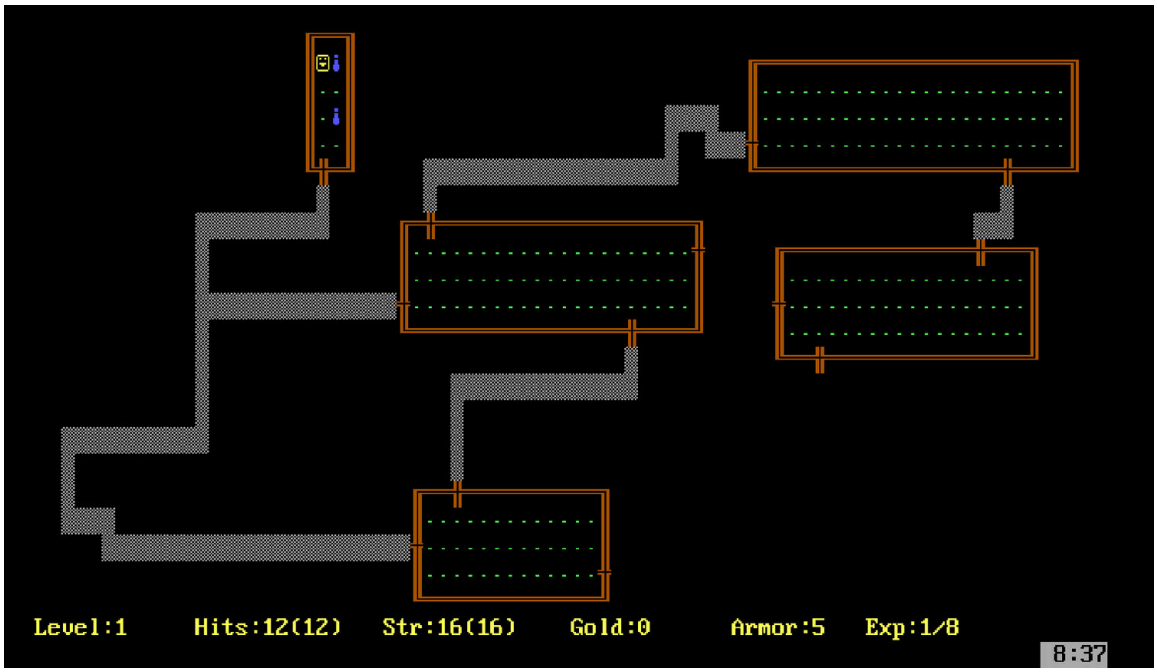
W niniejszym rozdziale omówiono podstawy proceduralnej generacji treści oraz scharakteryzowano gatunek gier roguelike. Przedstawiono ograniczenia dotychczas stosowanych metod oceny i analizy algorytmów, wskazując na potrzebę wprowadzenia zautomatyzowanych i porównywalnych miar. Zaprezentowano także przegląd wybranych algorytmów generacji wraz z opisem zasad ich działania.

2.1 Wprowadzenie do dziedziny proceduralnego generowania treści

Proceduralne generowanie treści stanowi jedną z kluczowych technik współczesnego przemysłu gier komputerowych. Umożliwia ono automatycznie tworzyć elementy rozgrywki bez bezpośredniej, ręcznej ingerencji projektantów, co znacząco obniża koszty produkcji oraz pozwala zwiększyć różnorodność wytwarzanych treści. Mechanizmy PCG znajdują zastosowanie w wielu aspektach tworzenia gier, począwszy od generowania tekstur, modeli czy fabuły, a skończywszy na dynamicznym konstruowaniu poziomów i całych światów.

Definicja 1. *Proceduralne generowanie treści to technika tworzenia zawartości gry za pomocą algorytmów komputerowych, często z wykorzystaniem elementów losowości, w celu automatyzacji procesu projektowania poziomów, obiektów, narracji lub innych elementów rozgrywki. Zawartość generowana w tym procesie musi być grywalna oraz dostosowana do ograniczeń i założeń projektowych danej gry [13].*

W literaturze rozróżnia się dwa zasadnicze tryby proceduralnego generowania treści: z wyprzedzeniem (ang. *offline*), na bieżąco (ang. *online*). Generowanie z wyprzedzeniem wykonywane jest przed uruchomieniem programu docelowego i obejmuje wytwarzanie większych porcji danych, które następnie zostają poddane weryfikacji jakościowej oraz ewentualnym poprawkom. Pozwala to korzystać z technik o dużej złożoności obliczeniowej



Rysunek 2.1: Zrzut ekranu przedstawiający fragment rozgrywki z gry *Rogue* (1980). Źródło: `https://store.steampowered.com/app/1443430/Rogue/`.

(np. wielokrotnych optymalizacji), które nie są możliwe do wykonania w trakcie działania programu. Generowanie na bieżąco zachodzi natomiast w trakcie działania programu docelowego i reaguje na aktualny stan systemu oraz działania użytkownika. Ten tryb wymaga bezwzględnego dotrzymania budżetów czasu i pamięci, deterministycznej powtarzalności wyników, mechanizmów strumieniowania danych oraz szybkiej oceny poprawności. W projektach powszechnie stosuje się również ujęcie hybrydowe: struktura wysokiego poziomu powstaje z wyprzedzeniem, natomiast uszczegółowienie i dostosowanie do kontekstu, na bieżąco [13].

2.2 Gry typu roguelike

Gatunek *roguelike* należy do kluczowych nurtów w projektowaniu gier komputerowych, zwłaszcza w kontekście generatywnych technik tworzenia zawartości. Pojęcie to wywodzi się z klasycznej gry *Rogue* z 1980 roku, autorstwa Michaela Toya, Glenna Wichmana i Kena Arnolda (rys. 2.1). Do jej cech wyróżniających należy: w pełni proceduralna generacja poziomów oraz mechanika trwałej śmierci postaci gracza (ang. *permadeath*), wiążąca się z koniecznością rozpoczęcia rozgrywki od nowa. Tytuł ten zapoczątkował nurt gier o zbliżonej strukturze i filozofii projektowania, które z czasem określono wspólnym mianem *roguelike* [16].

W literaturze przedmiotu i dyskursie środowiska twórców termin *roguelike* bywa doprecyzowywany za pomocą zestawu kryteriów, zwanego potocznie „Interpretacją Berliń-

Rysunek 2.2: Zrzut ekranu przedstawiający fragment rozgrywki z gry *Spelunky* (2008). Źródło: `https://store.steampowered.com/app/239350/Spelunky/`.

ską”[1]. Do najczęściej przywoływanych cech należą między innymi: rozgrywka turowa (ang. *turn-based gameplay*), reprezentacja świata w siatce dyskretnej, eksploracja labiryntowych struktur, wysoki poziom nieprzewidywalności wynikający z losowości, nacisk na zarządzanie zasobami oraz nieodwracalność decyzji [20]. Kluczowe jest jednak to, że losowość nie stanowi celu sama w sobie, lecz jest ona narzędziem do uzyskania wysokiej odtwarzalności (ang. *replayability*) i emergentnej złożoności interakcji. Można wiec wyróżnić kilka podstawowych cech gier z tego gatunku:

- **Rozgrywka turowa** — każdy ruch gracza koresponduje z ruchem świata gry, co sprzyja planowaniu i taktycznemu podejściu.
- **Proceduralna generacja poziomów** — układy są tworzone *na bieżąco*, co zwiększa różnorodność i unikalność doświadczeń.
- **Trwała śmierć postaci** — brak możliwości kontynuacji po porażce podnosi stawkę i wymusza ostrożne podejmowanie decyzji.
- **Eksploracja poziomów** — gracz przemierza wielopoziomowe, często labiryntowe struktury, wypełnione pułapkami, przeciwnikami oraz przedmiotami.

Gatunek *roguelike* przeszedł wyraźną ewolucję: od prostych, tekstowych interfejsów do złożonych, trójwymiarowych środowisk. Współczesne realizacje łączą kanoniczne rozwiązania charakterystyczne dla tego typu gier z nowoczesnymi mechanikami takimi jak:

[1]Interpretacja ta ma charakter deskryptywny i historyczny, a nie normatywny; poszczególne produkcje spełniają ją w różnym stopniu. Doprowadziło to do wyodrębnienia podgatunków, takich jak *roguelite* czy *dungeon crawler*, zachowujących jedynie wybrane elementy pierwowzoru.

Rysunek 2.3: Zrzut ekranu przedstawiający fragment rozgrywki z gry *The Binding of Isaac* (2011). Źródło: `https://store.steampowered.com/app/113200/The_Binding_of_Isaac/`.

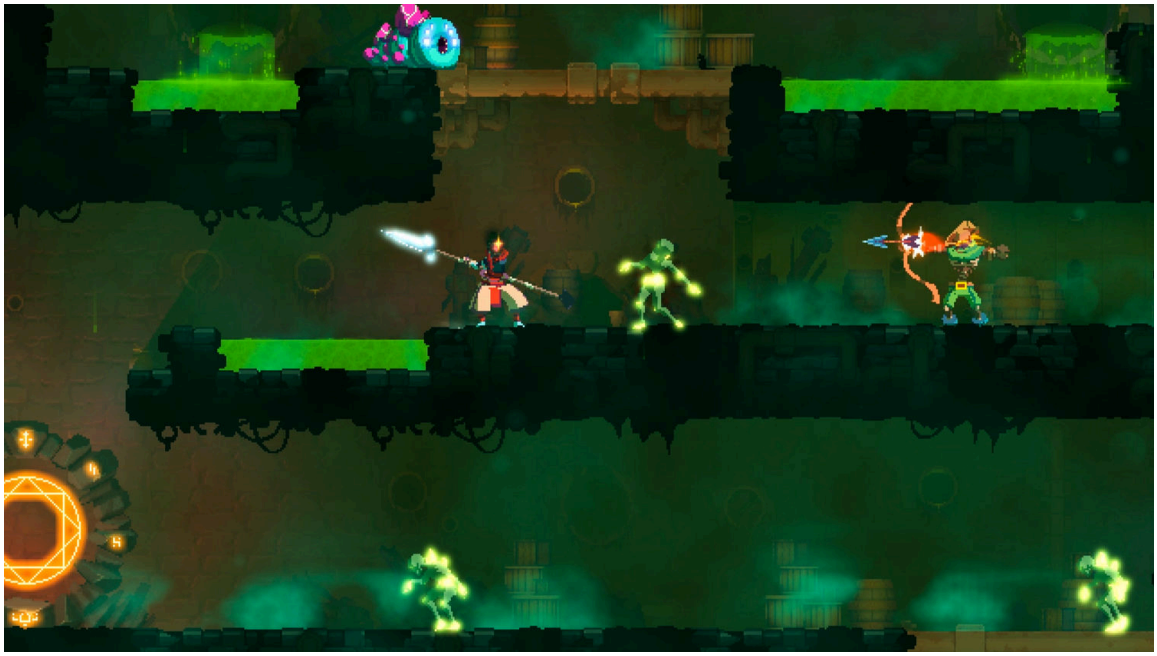
rozwój postaci, systemy wytwarzania przedmiotów, rozbudowana narracją, gromadzeniee zasobów oraz interakcjami z bohaterami niezależnymi (ang. *non-player character*, NPC). Wymienione czynniki przekładają się na wielowarstwowe, złożone doświadczenie rozgrywki. Mimo rozwoju technologicznego, techniki proceduralnego generowania poziomów pozostają fundamentem projektowania gier, umożliwiając tworzenie bogatych, zróżnicowanych światów, które zachęcają do eksploracji i ponownych powrotów.

Przedstawicielami współczesnego nurtu są takie tytuły jak "*The Binding of Isaac*"(rys. 2.3), "*Spelunky*"(rys. 2.2), "*Dead Cells*"(rys. 2.4) czy "*Hades*"(rys. 2.5). Każda z tych gier wprowadza unikalne mechaniki i stylistykę, jednocześnie zachowując kluczowe elementy gatunku. Ich sukces komercyjny jak również pozytywne opinie wśród krytyków świadczą o trwałej atrakcyjności koncepcji *roguelike* oraz o skuteczności proceduralnego generowania treści jako narzędzia projektowego.

Roguelike jako kontekst badań nad proceduralnym generowaniem treści

W niniejszej pracy gry typu *roguelike* traktowane są jako modelowe środowisko do weryfikacji algorytmów proceduralnego generowania treści. W szczególności pomocne są następujące cechy tego gatunku:

- Wysoka częstotliwość losowego tworzenia poziomów wymusza na algorytmach nie tylko dużą wydajność obliczeniową, lecz także spełnienie podstawowych wymagań strukturalnych, takich jak spójność poziomu oraz osiągalność punktów specjalnych.



Rysunek 2.4: Zrzut ekranu przedstawiający fragment rozgrywki z gry *Dead Cells* (2018). Źródło: `https://store.steampowered.com/app/588650/Dead_Cells/`.

Rysunek 2.5: Zrzut ekranu przedstawiający fragment rozgrywki z gry *Hades* (2020). Źródło: `https://store.steampowered.com/app/1145360/Hades/`.

- Zmienność kolejnych instancji poziomów umożliwia prowadzenie licznych prób empirycznych z wykorzystaniem automatycznych symulacji gracza, co sprzyja szacowaniu miar jakości rozgrywki.
- Zróżnicowanie typów struktur występujących w poziomach (np. labirynty, jaskinie, zespoły komnat) motywuje analizę i porównanie odmiennych klas metod i algorytmów generujących, co pozwala na wyciąganie bardziej ogólnych wniosków.
- Jasno określony cel rozgrywki, polegający typowo na pozyskaniu przedmiotu warunkującego postęp i dotarciu do wyjścia. Pozwala to ujednolicić strategie w symulacjach zachowań oraz precyzyjnie zdefiniować miary sukcesu.

Generowanie poziomów można więc traktować jako powtarzalny, uruchamiany przy każdej nowej rozgrywce proces wytwórczy. Zamiast projektować pojedynczą, statyczną mapę, system za każdym razem konstruuje nową konfigurację pomieszczeń i korytarzy, tak aby kolejne sesje różniły się od siebie, a zarazem pozostawały zrozumiałe i uczciwe wobec gracza. Celem generowania nie jest pełna, chaotyczna losowość, lecz systematyczne wytwarzanie sensownych wyzwań i sytuacji decyzyjnych, które zapewniają wysoką powtarzalność i unikalność doświadczenia.

Proces ten obejmuje planowanie struktury poziomu, przypisanie ról funkcjonalnych (np. start, cel, skarb, wyzwanie, przeciwnik) oraz implementacje mechanizmów progresji (np. drzwi wymagające klucza). Całość jest sterowana zestawem parametrów oraz ziarnem losowości (ang. *seed*), co umożliwia jednoczesne zarządzanie różnorodnością i odtwarzalnością wyników w celach projektowych i testowych. Każda instancja poziomu podlega automatycznej walidacji względem minimalnych kryteriów jakości. W szczególności sprawdzana jest spójność topologiczna mapy, istnienie ścieżki od punktu startowego do celu lub innych specyficznych założeń dla konkretnego tytułu lub środowiska. W przypadku naruszenia któregokolwiek warunku uruchamiane są proste procedury naprawcze (np. dołożenie przejścia, rekonstrukcja fragmentu mapy) lub dany układ zostaje odrzucony i wylosowany ponownie.

Struktura poziomu oraz stopień jego eksplorowalności bezpośrednio warunkują możliwość realizacji celu rozgrywki, stanowiąc tym samym kluczowe kryteria oceny jakości wygenerowanych poziomów. Podstawowym walorem proceduralnego generowania treści w grach *roguelike* jest wysoka powtarzalność rozgrywki połączona z nieprzewidywalnością napotykanych sytuacji. Każda sesja znacząco różni się od poprzednich, a zastosowane algorytmy determinują topologię poziomu, rozmieszczenie przeciwników, zasobów i elementów fabularnych, przez co bezpośrednio wpływają na obciążenie poznawcze i zaangażowanie gracza [14]. Jakość procesu generacji przekłada się na takie własności, jak różnorodność środowiska, adekwatny poziom wyzwania, spójność układów czy stopień immersji. Z tych względów dobór oraz ewaluacja metod generujących stanowi integralny element zarówno projektowania, jak i weryfikacji gier osadzonych w tej konwencji.

2.3 Sposoby ewaluacji jakości generowanych poziomów

Dotychczasowe badania w obszarze oceny jakości grywalności generowanych poziomów w dużej mierze opierają się na eksperymentach z udziałem ekspertów lub graczy. W tego typu podejściach jakość poziomów oceniana jest na podstawie subiektywnych rankingów, co czyni wyniki badań silnie zależnymi od kontekstu eksperymentu, doboru uczestników oraz przebiegu testów. W konsekwencji utrudnia to porównywanie rezultatów pomiędzy różnymi badaniami i ogranicza możliwość formułowania uogólnionych wniosków. W badaniu Rodriguesa [10] podstawowymi wskaźnikami były ankiety przeprowadzane wśród graczy (m.in. odczuwana przyjemność rozgrywki, ciekawość, chęć powrotu), co silnie wiąże wnioski z kontekstem konkretnej gry i składem próby. W badaniu [5] wykorzystano ankietową ocenę lojalnościową (ang. *Net Promoter Score*) przy niewielkiej liczbie uczestników, wyniki w dużej mierze zależały od doboru respondentów, ich liczebności oraz konstrukcji testu. W obu przypadkach dominacja miar subiektywnych utrudnia porównywalność rezultatów między badaniami i ogranicza możliwość formułowania uogólnionych wniosków. Kompleksowa analiza przeprowadzona przez Withington, Cook i Tokarchuk [15] wskazuje, że współczesne badania nad proceduralnym generowaniem poziomów napotykają fundamentalne problemy w zakresie metodologii ewaluacji. Wspomniani autorzy, dokonując w 2024 roku przeglądu 86 publikacji opublikowanych w ciągu ostatnich pięciu lat, zidentyfikowali kluczowe niedoskonałości w obecnie stosowanych praktykach badawczych, które uzasadniają potrzebę opracowania zaawansowanych, zautomatyzowanych systemów oceny jakości generowanych treści. Wyniki badania ujawniają, że ponad połowa analizowanych prac (47 z 86) opiera się na całkowicie oryginalnych domenach testowych, co uniemożliwia bezpośrednie porównanie uzyskanych rezultatów między różnymi badaniami. Taka fragmentacja metodologiczna stanowi istotną barierę w rozwoju dziedziny, gdyż utrudnia ocenę rzeczywistego postępu w zakresie algorytmów generacji poziomów. W kontekście niniejszej pracy problem ten dodatkowo podkreśla konieczność opracowania zunifikowanego narzędzia testowego, które umożliwi systematyczną i rzetelną ewaluację różnych podejść w kontrolowanych warunkach eksperymentalnych.

Przykładem podejścia opartego na analizie struktury poziomów jest praca przeprowadzona przez Liapis, Yannakakis i Togelius [9], w której zaproponowano zestaw metryk topologicznych pozwalających na obiektywną ewaluację jakości map w różnych gatunkach gier. Autorzy modelują poziom jako siatkę pól przechodnich i nieprzechodnich, a następnie definiują sześć miar opisujących trzy kluczowe aspekty projektowania: kontrolę obszaru (mierzoną m.in. poprzez odległości i pojęcie względnego bezpieczeństwa pól względem punktów strategicznych), eksplorację (określaną za pomocą algorytmu "*flood fill*"wyznaczającego pokrycie mapy) oraz balans (oceniany poprzez symetrię i równowagę w dostępie do zasobów i obszarów). Dodatkowo wprowadzono warunek spójności topolo-

gicznej jako kryterium konieczne do uznania poziomu za grywalny, eliminujący przypadki izolowanych fragmentów mapy. Proponowane metryki zostały zastosowane w grach strategicznych czasu rzeczywistego oraz w grach typu roguelike, a wyniki wykazały, że mogą one służyć jako uniwersalne narzędzie do systematycznej i rzetelnej oceny poziomów, niezależnie od gatunku czy specyfiki testowanej domeny.

W badaniu autorstwa Michael Beukman [11] pojawiła się propozycja wykorzystania symulacyjnych metryk opartych na działaniach agentów sztucznej inteligencji poruszających się po poziomach z wykorzystaniem algorytmów wyszukujących najkrótsze ścieżki. Autorzy tej publikacji zaproponowali dwie miary, które analizują poziomy gier z perspektywy zachowania autonomicznego agenta. Pierwsza z nich, metryka różnorodności, oblicza odległość między sekwencjami działań wykonanych przez agenta w różnych poziomach, co pozwala ocenić zróżnicowanie wymaganych strategii rozwiązywania. Druga, metryka trudności, określa stopień eksploracji drzewa wyszukiwania, niezbędnej do znalezienia ścieżki do celu, co odzwierciedla obecność pułapek i złożoność układu przestrzennego. Analiza przeprowadzona przez autorów wskazuje, że opisywane podejście jest mniej wrażliwe na czysto wizualne różnice w układach oraz skalę poziomów, a tym samym trafniej opisuje aspekty związane z grywalnością. Metody te wyznaczają kierunek ku obiektywniejszej ocenie poziomów, jednak ich ograniczona walidacja oraz brak szerokiego zastosowania w badaniach stanowi barierę przed uznaniem ich za standard.

Podsumowując, należy zauważyć, że aktualnie stosowane w literaturze metody ewaluacji generowanych poziomów, zarówno te bazujące na subiektywnych opiniach uczestników badań, jak i te odwołujące się do różnego rodzaju metryk strukturalnych opisujących geometrię i topologię map, okazują się w praktyce niewystarczające do przeprowadzenia pełnej, rzetelnej i możliwie obiektywnej oceny jakości treści tworzonych w sposób proceduralny. Ich ograniczenia przejawiają się przede wszystkim w tym, że albo odzwierciedlają one jedynie indywidualne doświadczenia i preferencje konkretnych graczy, zależne od kontekstu eksperymentu, albo sprowadzają analizę do relatywnie prostych parametrów przestrzennych, które nie oddają całego bogactwa i złożoności interakcji zachodzących podczas rozgrywki. W konsekwencji trudno jest na ich podstawie formułować uogólnione i porównywalne wnioski, a tym bardziej wyznaczać kierunki dalszego rozwoju algorytmów generatywnych. Brakuje zatem zestawu uniwersalnych metryk, które w sposób kompleksowy uwzględniałyby zarówno strukturalny i topologiczny wymiar poziomów, jak i ich wpływ na szeroko rozumiane doświadczenia, emocje oraz zaangażowanie graczy, pozwalając na spójną ewaluację różnych podejść badawczych w ramach jednolitych ram metodologicznych.

2.4 Przegląd algorytmów proceduralnego generowania poziomów

W niniejszej pracy przez „poziom” rozumie się dwuwymiarową siatkę prostokątną, zdolną do reprezentowania różnych typów struktur, takich jak labirynty, jaskinie czy układy komnatowo-korytarzowe. Przyjęto następujące założenia dotyczące reprezentacji poziomu:

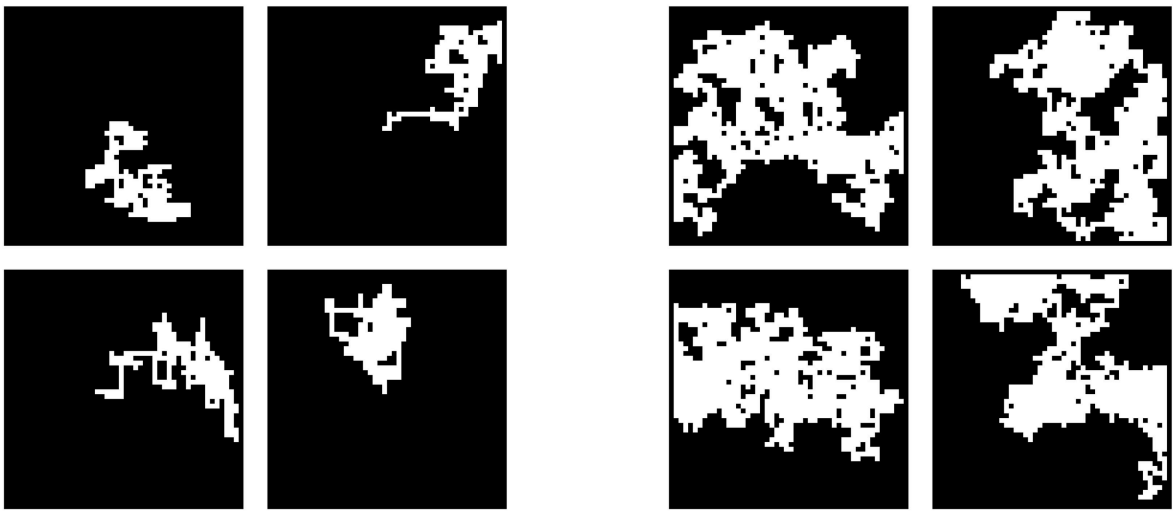
- komórki planszy mogą przyjmować dwa stany: ściana (nieprzekraczalna) lub podłoga (przejście),
- plansza ma określone wymiary (szerokość i wysokość),
- krawędź planszy traktujemy jako nieprzekraczalną granicę,
- określone komórki pełnią dodatkową rolę punktów specjalnych, takich jak start, wyjście czy klucz do wyjścia,
- istnieje ścieżka przejść między startem a wyjściem.

Przedstawione algorytmy zostały dobrane w taki sposób, aby reprezentowały różne paradygmaty konstrukcji map: podejście agentowe, metody konstruktywno-hierarchiczne, reguły lokalne, modelowanie pól ciągłych z progowaniem oraz wzrost klastrowy ograniczony dyfuzją. Dobór ten uzasadnia ugruntowana obecność tych metod w literaturze i praktyce gatunku *roguelike* [13], komplementarność generowanych stylów geometrycznych, prostota implementacyjna oraz przejrzysta parametryzacja umożliwiająca kontrolę nad wynikiem. Z tego względu pominięto algorytmy oparte na modelach uczonych, algorytmy ewolucyjne oraz sieci neuronowe, które wymagają złożonego procesu trenowania lub skomplikowanych założeń i nie są one jeszcze powszechnie stosowane w praktyce. Wykluczono również rozwiązania bazujące na uprzednio przygotowanych przez twórcę poziomu szablonów struktur, gdyż nie są one w pełni proceduralne i nie spełniają założeń pracy.

2.4.1 Algorytm *Pijany Spacer*

Algorytm *Pijany Spacer* (ang. *Drunkard's Walk*) [2] jest reprezentantem kategorii algorytmów podejść agentowych i należy do najprostszych metod stochastycznych stosowanych w generowaniu poziomów w grach komputerowych. Jego istota opiera się na symulacji jednego lub wielu losowych wędrowców, którzy poruszają się po poziomie w nieprzewidywalny sposób, stopniowo odsłaniając kolejne przejścia i korytarze. Nazwa metody nawiązuje do chaotycznego toru ruchu przypominającego chód nietrzeźwego człowieka.

Ze względu na prostotę implementacyjną oraz niskie wymagania obliczeniowe, algorytm ten jest szeroko wykorzystywany do konstruowania nieregularnych układów kory-



Rysunek 2.6: Dwie przykładowe wizualizacje działania algorytmu *Pijany Spacer* dla różnych parametrów początkowych.

tarzy i labiryntów a w literaturze poświęconej PCG algorytm *Pijany Spacer* bywa klasyfikowany jako metoda „chaotyczna" [17]. Jego podstawy wywodzą się z klasycznego zagadnienia matematycznego znanego jako losowe błądzenie (ang. *random walk*), które od wielu dekad stanowi obiekt intensywnych badań w takich dziedzinach jak chemia [7], fizyka [19] czy biologia [12]. Modele losowego błądzenia znajdują zastosowanie między innymi przy opisie procesów dyfuzji, migracji komórek w tkankach oraz ruchu cząstek w zawiesinach.

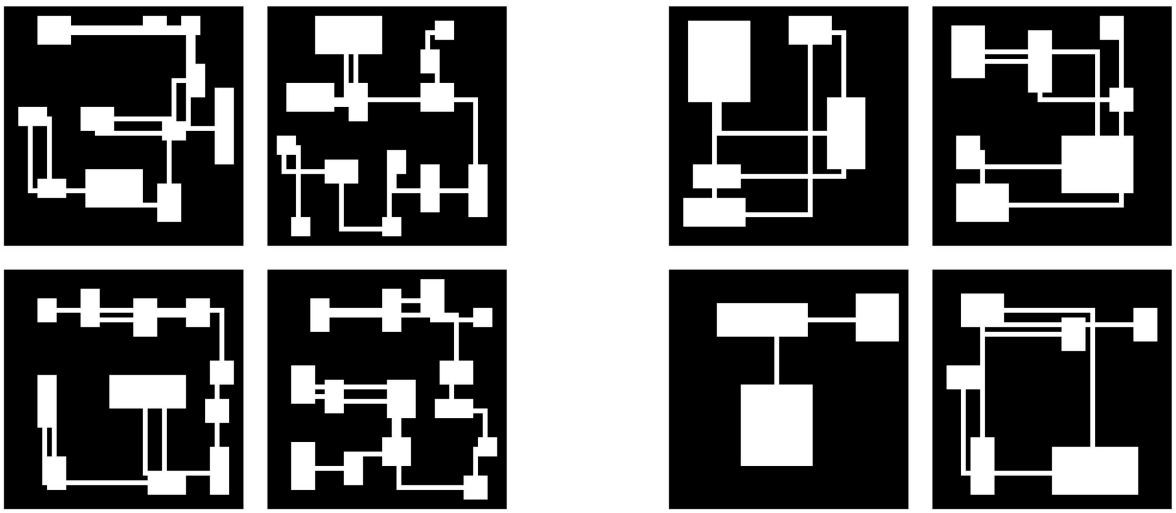
Zasada działania. Na początku cała plansza zostaje wypełniona ścianami. Następnie w centrum umieszcza się jednego lub kilku wędrowców. W kolejnych iteracjach poruszają się oni losowo w jednym z czterech kierunków. Każdy punkt na planszy odwiedzony przez wędrowca zostaje oznaczony jako podłoga. Po odpowiedniej liczbie kroków rezultatem działania algorytmu jest mapa o nieregularnym, organicznym układzie korytarzy, przypominająca naturalnie powstałe jaskinie lub tunelowe systemy (rys. 2.6).
Przykładowy przebieg działania algorytmu *Pijany Spacer*:

1. Dla każdego wędrowcy wybierz kierunek ruchu uwzględniając opcjonalną preferencje ruchu w określonym kierunku.

2. Przesuń wędrowców, oznaczając aktualną pozycję jako podłoge.

3. Na podstawie prawdopodobieństwa powrotu zdecyduj, czy kontynuować, czy cofnąć danego wędrowce na poprzednią pozycję.

4. Powtarzaj proces aż do osiągnięcia ustalonej liczby kroków lub wypełnienia wymaganej części siatki.



Rysunek 2.7: Dwie przykładowe wizualizacje działania algorytmu *BSP* dla różnych wartości parametrów wejściowych.

5. Wynikiem działania algorytmu jest przekształcona siatka reprezentująca strukturę poziomu.

2.4.2 Algorytm *Binarnego Podziału Przestrzeni*

Algorytm *Binarnego Podziału Przestrzeni* (ang. *Binary Space Partitioning, BSP*) [18] [3] należy do metod konstruktywno-hierarchicznej dekompozycji przestrzeni stosowanych w proceduralnej generacji poziomów. Jego idea polega na rekurencyjnym dzieleniu prostokątnej planszy na mniejsze regiony, a następnie tworzeniu komnat, które łączy się korytarzami. Technika wywodzi się z grafiki komputerowej [22] i znalazła szerokie zastosowanie w grach typu *roguelike* jako sposób na uzyskanie spójnych, czytelnych układów „komnaty–korytarze" o kontrolowalnych rozmiarach i gęstości zabudowy (rys. 2.7).

W przeciwieństwie do podejść agentowych, które opierają się na lokalnych decyzjach, *BSP* traktuje przestrzeń globalnie i pozwala zaplanować całą strukturę poziomu odgórnie. Jak zauważają Shaker, Togelius i Liapis: „Algorytm *BSP* gwarantuje, że żadne dwie komnaty nie będą się nakładały, a cały loch zyskuje uporządkowany i spójny wygląd" [13]. Dzięki temu metoda ta zapewnia zarówno kontrolę nad rozkładem pomieszczeń, jak i przejrzystością układu.

Zasada działania. Definiuje się obszar roboczy wewnątrz planszy zachowując margines od krawędzi i uruchamia procedurę rekurencyjnego podziału. Na każdym etapie wybierany jest kierunek cięcia z preferencją dla dłuższego wymiaru regionu. Gdy wymiary są porównywalne, kierunek losuje się. Podział jest dozwolony tylko wtedy, gdy oba powstałe podobszary spełnią minimalne wymagania rozmiaru oraz zachowany zostanie zadany mar-

gines od miejsca cięcia. Rekursja kończy się po osiągnięciu maksymalnej głębokości lub gdy region staje się zbyt mały, by sensownie go dalej dzielić. W końcowych podobszarach powstają komnaty których szerokość i wysokość losuje się w obrębie minimalne i maksymalnego rozmiaru pokoju.

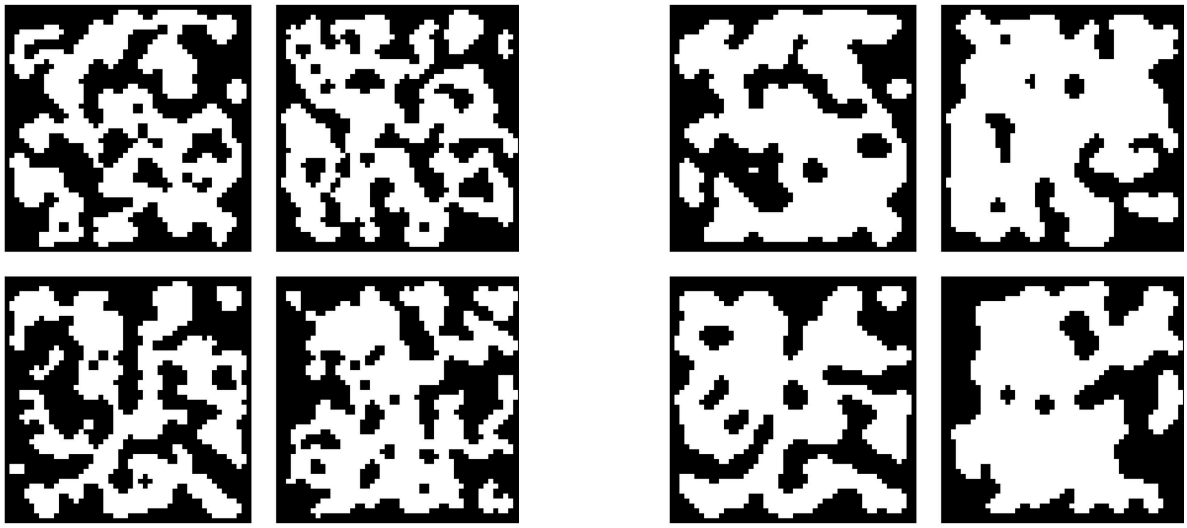
Po wygenerowaniu wszystkich komnat ich wnętrza są oznaczane jako podłoga. Następnie komnaty łączy się korytarzami, łącząc kolejno centra geometryczne sąsiadujących na liście komnat. Korytarz ma kształt litery „L", a każda komórka na trasie jest oznaczana jako przejście. Taki sposób łączenia zapewnia spójność mapy i prostotę implementacji, a jednocześnie pozostawia miejsce na typowe rozszerzenia (np. dodatkowe połączenia między komnatami, losowe „złamania" korytarzy, drzwi, poszerzanie korytarzy).
Przykładowy przebieg działania algorytmu *BSP*:

1. Zainicjalizuj całą przestrzeń gry jako jeden region.
2. Rekurencyjnie dziel regiony na dwa podregiony do momentu spełnienia warunku stopu.
3. Wygeneruj pokój w każdym końcowym podobszarze, uwzględniając ograniczenia rozmiaru i położenia.
4. Połącz pokoje korytarzami zgodnie z kolejnością podziału i geometryczną odległością.
5. Wynikiem działania algorytmu jest przekształcona siatka reprezentująca strukturę poziomu.

2.4.3 Algorytm *Automatu Komórkowego*

Algorytmy oparte na Automatach Komórkowych (ang. *Cellular Automaton, CA*) należą do klasy metod lokalnych [6], w których złożona, globalna struktura poziomu wyłania się z prostych i jednocześnie stosowanych reguł sąsiedztwa. Dzięki temu możliwe jest generowanie złożonych układów przestrzennych przy użyciu relatywnie prostych mechanizmów obliczeniowych. Automaty komórkowe szczególnie dobrze sprawdzają się w zadaniach związanych z modelowaniem procesów naturalnych, takich jak powstawanie jaskiń, układów tuneli czy nieregularnych struktur terenowych w grach komputerowych (rys. 2.8).

Zasada działania. Każda komórka otrzymuje początkowy stan (ściana lub podłoga) zgodnie z ustalonym prawdopodobieństwem. Następnie przeprowadza się kilka iteracji jednoczesnej aktualizacji całej planszy na podstawie wybranej reguły progowej. Stosuje się sąsiedztwo Moore'a [4] (obejmujące osiem komórek otaczających komórkę centralną), które pozwala uchwycić zarówno bezpośrednich, jak i diagonalnych sąsiadów. W każdej



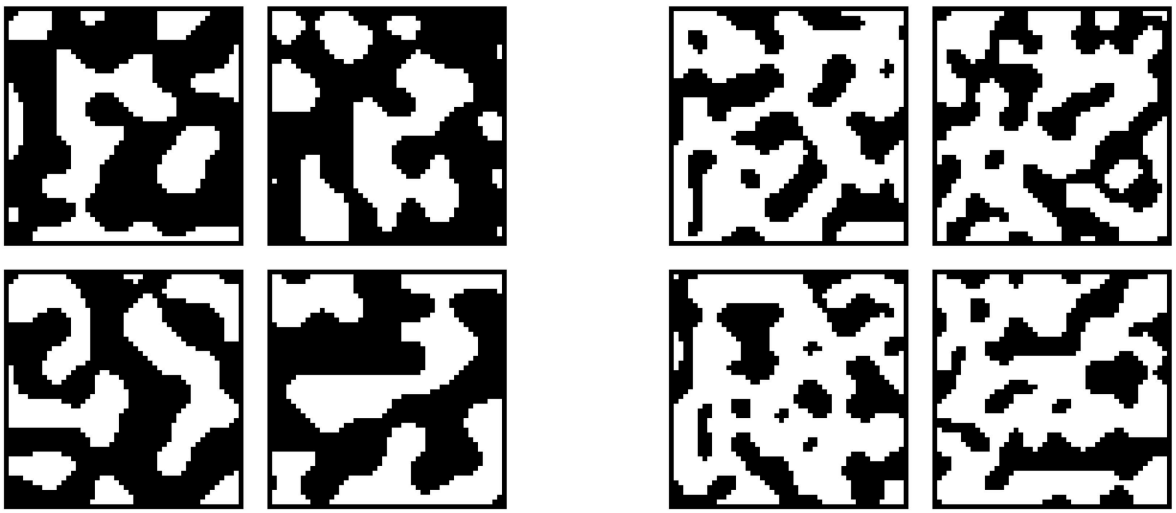
Rysunek 2.8: Dwie przykładowe wizualizacje działania algorytmu *Automatów Komórkowych* dla różnych parametrów.

iteracji sprawdzana jest liczba komórek ścian w otoczeniu danej komórki. Jeśli spełniony zostanie określony warunek progowy (np. większość sąsiadów to ściany), komórka w następnej iteracji przyjmuje stan ściany, w przeciwnym przypadku pozostaje przestrzenią pustą. Dodatkowo, w celu naturalnego domknięcia poziomu, przyjmuje się, że komórki znajdujące się poza granicami planszy traktowane są jako ściany. Takie założenie sprawia, że krawędzie generowanego poziomu tworzą spójną, zamkniętą strukturę.
Przykładowy przebieg działania algorytmu *Automatów Komórkowych*:

1. Dla każdej komórki wylosuj stan, ustawiając ją jako ścianę lub podłogę.
2. Dla każdej komórki policz liczbę sąsiadów będących ścianami (8-kierunkowo).
3. Jeśli liczba ścian jest większa od zadanego progu powstanie ściany ustaw komórkę tam ścianę.
4. Jeśli liczba ścian jest mniejsza od zadanego progu powstania podłogi ustaw tam podłogę.
5. Po przejściu przez wszystkie iteracje otrzymujemy wygładzoną siatkę, która stanowi strukturę poziomu.

2.4.4 Algorytm *Szum Perlina*

Algorytmy oparte na Szumie Perlina (ang. *Perlin Noise*) [8] generują ciągłe, gładkie pola wartości, które po progowaniu zamieniają się w mapy ścian i przejść. Szum Perlina świetnie sprawdza się przy tworzeniu „organicznych” jaskiń i tuneli zamiast losowych,



Rysunek 2.9: Dwie przykładowe wizualizacje działania algorytmu *Szum Perlina* dla różnych wartości.

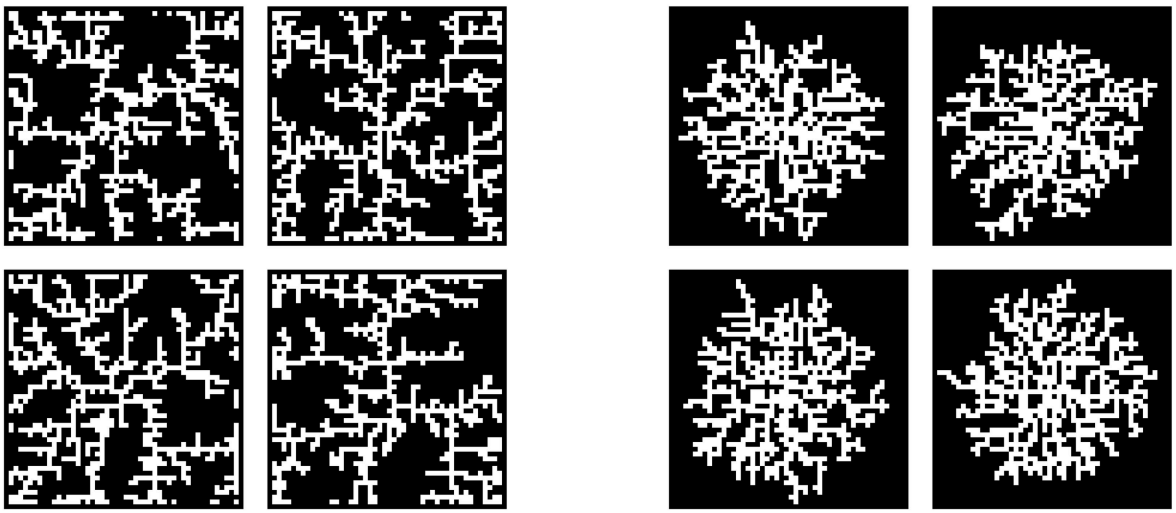
postrzępionych przejść czy regularnych układów komnat, otrzymujemy spójne wyspy i korytarze o płynnych krawędziach. Dzięki niewielkiej liczbie parametrów metoda ta jest prosta do strojenia i powtarzalna (rys. 2.9).

Zasada działania. Dla każdej komórki siatki próbkujemy dwuwymiarowy szum Perlina i wynikową wartość porównujemy z progiem przepuszczalności. Jeśli wartość jest wyższa niż zadany próg, komórka staje się podłogą, w przeciwnym wypadku pozostaje ścianą. Aby uniknąć powtarzalnych artefaktów, do współrzędnych przykładamy przesunięcie (ang. *offset*) wynikające z ziarna generatora. Skalę i poziom szczegółowości kontrolują parametry skali, częstotliwości oraz amplitudy.
Przykładowy przebieg działania algorytmu *Szum Perlina*:

1. Każdą komórkę ustaw jako ścianę.
2. Dla każdej komórki policz odpowiadającą jej wartość szumu Perlina.
3. Oznacz komórkę jako podłogę, gdy wyznaczona wartość przekracza zadany próg, w przeciwnym razie pozostaw tam ścianę.

2.4.5 Algorytm *Agregacji Ograniczonej Dyfuzją*

Agregacja Ograniczona Dyfuzją (ang. *Diffusion-Limited Aggregation*, DLA) stanowi model stochastyczny opisujący proces powstawania rozgałęzionych struktur poprzez przyłączanie cząstek podlegających losowemu ruchowi do rozwijającego się klastra. W literaturze przedmiotu metoda ta wykorzystywana jest do badania zjawisk wzrostu kryształów,



Rysunek 2.10: Dwie przykładowe wizualizacje działania algorytmu *DLA* dla różnych wartości parametrów.

korozji czy formacji dendrytycznych w układach fizycznych [23]. Jej szczególną cechą jest zdolność do generowania wysoce nieregularnych, rozgałęzionych struktur, które odzwierciedlają procesy naturalnej dyfuzji (rys. 2.10).

Algorytm *DLA* pozwala uzyskać układy korytarzy i komnat o charakterze silnie dendrytycznym. Różnią się one istotnie od struktur tworzonych przy pomocy automatów komórkowych, jak i od regularnych układów komnat otrzymywanych poprzez podziały przestrzeni. Dzięki prostej strukturze obliczeniowej, a jednocześnie wysokiemu potencjałowi generowania różnorodnych form, metoda ta znajduje zastosowanie w projektowaniu poziomów o zróżnicowanej topologii i nieliniowym przebiegu ścieżek eksploracyjnych.

Zasada działania. W poziomie umieszcza się pojedynczy „zarodek" klastra (komórkę podłogi). W kolejnych iteracjach uruchamia się wędrowców startujących na okręgu o promieniu nieco większym niż dotychczasowy zasięg klastra. Każdy wędrowiec porusza się losowo w jednym z ośmiu kierunków, przy czym kierunki mogą być uprzywilejowane w stronę środka planszy. Jeżeli wędrowiec dotknie klastra (sąsiedztwo 4-kierunkowe), z pewnym prawdopodobieństwem przylega do niego i staje się częścią struktury. Aby zapobiec „ucieczce" wędrowców zbyt daleko, stosuje się promień usunięcia, poza którym wędrowiec jest porzucany i zastępowany kolejnym.
Przykładowy przebieg działania algorytmu *DLA*:

1. Wypełnij planszę ścianami i ustaw w centrum pojedynczy zarodek klastra.
2. Umieść wędrowca na okręgu startowym otaczającym klaster.
3. Wykonuj kroki losowe w jednym z 8 kierunków, preferując ruch w stronę centrum.

4. Jeśli wędrowiec sąsiaduje z klastrem, dołącz go do struktury.
5. Jeśli wędrowiec oddali się poza promień usunięcia, porzuć go i rozpocznij nową próbę.
6. Powtarzaj proces aż klaster osiągnie zadaną wielkość.
7. Wynikiem działania algorytmu jest siatka z dendrytycznym, rozgałęzionym układem korytarzy.

Rozdział 3

Opis środowiska badawczego

Opracowany system został zaprojektowany jako zintegrowane środowisko badawcze, umożliwiające pełną automatyzację procesu generowania, testowania oraz analizy poziomów. Cały łańcuch działania, od generowania poziomów, poprzez obliczanie metryk i uruchamianie symulacji, aż po agregację oraz eksport wyników który realizowany jest według z góry zdefiniowanego scenariusza. Taki układ minimalizuje ingerencję człowieka w trakcie pomiaru, ogranicza ryzyko błędów operacyjnych i sprzyja standaryzacji przebiegu badań, co wprost przekłada się na większą wiarygodność uzyskiwanych rezultatów. Replikowalność zapewniono poprzez deterministyczną kolejność uruchamiania konfiguracji, ścisłą kontrolę źródeł losowości, w tym stałe ustawienia generatorów liczb pseudolosowych, oraz jednolite procedury agregacji i raportowania. Zastosowanie spójnych konwencji prezentacji niepewności umożliwia odtworzenie analiz oraz porównywanie wyników między seriami eksperymentów. Jako środowisko implementacyjne wybrano silnik *Unity*, który charakteryzuje się dużą elastycznością w zakresie prototypowania i wizualizacji, co umożliwia szybkie opracowywanie zarówno generatorów poziomów, jak i ich reprezentacji w dwóch oraz trzech wymiarach [21].

Architektura środowiska została zaprojektowana z myślą o skalowalności. Procedury obliczeniowe i organizacja pracy pozwalają na bezpieczne zwiększanie liczby konfiguracji, iteracji oraz przebiegów symulacji bez konieczności zmiany paradygmatu badawczego. Dzięki temu możliwe jest dostosowanie zakresu eksperymentu do dostępnych zasobów obliczeniowych i czasu, bez ryzyka utraty spójności wyników.

Struktura środowiska badawczego oddziela warstwę generacji od warstwy wykonującej testy. Rozdzielenie tych funkcji wzmacnia bezstronność porównań między algorytmami i ułatwia niezależny rozwój komponentów. Dzięki temu implementacje testów mogą być rozszerzane bez ingerencji w mechanizmy generacji, a wtórna analiza statystyczna przebiega na ujednoliconych, eksportowanych zbiorach danych, zgodnie z zasadą rozdziału pozyskiwania i interpretacji wyników. Tak zaprojektowany łańcuch przetwarzania zapewnia spójność danych między warstwami, możliwość równoległego przetwarzania dużych prób oraz jasny podział ról między wytwarzaniem, oceną i interpretacją wyników.

3.1 Architektura i przepływ danych

Środowisko badawcze ma budowę warstwową, co umożliwia rozdzielenie odpowiedzialności oraz zachowanie spójnego, powtarzalnego przepływu informacji między komponentami.

Generacja poziomów. Warstwa generacyjna wytwarza reprezentację poziomu w postaci siatki wraz z węzłami specjalnymi *(S/K/E)* dla zadanej konfiguracji parametrów i ziarna losowości.

Testy strukturalne i testy grywalności. Nad wygenerowaną siatką obliczany jest zestaw wskaźników opisujących własności topologiczne oraz wskaźniki grywalności. Zestaw testów jest ujednolicony pod względem sposobu uruchamiania i raportowania, co umożliwia agregację wyników w skali wielu konfiguracji.

Symulacja gracza. Niezależna warstwa symulacyjna uruchamia uproszczonego agenta eksplorującego poziom. Metryki grywalności liczone przez symulację gracza uzupełniają informacje strukturalne o wymiar behawioralny.

Orkiestracja i współbieżność. Orkiestrator zarządza planem eksperymentu: wyznacza iloczyn kartezjański parametrów, harmonogramuje partie, kontroluje postęp i koordynuje współbieżne uruchamianie zadań. Dzięki temu możliwe jest skalowanie eksperymentów przy zachowaniu powtarzalności procedur.

Eksport danych. Zagregowane wyniki są serializowane do ustrukturyzowanego formatu wraz z metadanymi o konfiguracji i rozmiarze próby. Warstwa ta stanowi stały punkt styku między częścią eksperymentalną a analizą statystyczną.

Wtórna analiza. Dane eksportowane z silnika eksperymentalnego są ładowane do środowiska analitycznego w celu przeprowadzenia analiz przekrojowych, korelacyjnych oraz wizualizacji. Rozdzielenie pozyskiwania i interpretacji danych sprzyja transparentności metod i replikowalności.

Generacja poziomów i ewaluacja

Model poziomu

Podstawową jednostką analizy jest instancja poziomu reprezentowana przez prostokątną siatkę pól, z rozróżnieniem na pola przechodnie (podłogi) i nieprzechodnie (ściany).

W strukturze tej wyróżnia się trzy węzły specjalne: punkt startowy (S), położenie klucza (K) oraz wyjście (E).

Metryki i testy jakości poziomów

Zestaw ewaluacyjny podzielono na dwie klasy, które komplementarnie opisują jakość poziomu:

1. **Metryki strukturalne** oceniają geometryczno–topologiczne własności siatki, niezależnie od konkretnych położeń *(S/K/E)*. Ich rola polega na uchwyceniu „kształtu" przestrzeni gry w sposób niezależny od scenariusza rozgrywki.

2. **Metryki grywalności** wykorzystują relacje pomiędzy punktami specjalnymi. Metryki te stanowią pomost pomiędzy geometrią poziomu a przewidywanym doświadczeniem gracza.

Powtarzalność i kontrola losowości

Aby zapewnić rzetelność porównań, środowisko rozdziela wpływ geometrii od wpływu położeń węzłów specjalnych:

- **Testy strukturalne** obliczane są jednokrotnie na instancji poziomu, ponieważ nie zależą od rozmieszczenia *(S/K/E)*.

- **Testy wrażliwe na relokację punktów specjalnych** wykonywane są wielokrotnie w ramach iteracji, w których następuje przetasowanie położeń punktów specjalnych; wyniki są następnie agregowane co stabilizuje estymację wskaźników zależnych od scenariusza przejścia. Przy agregacji uwzględniane są wszystkie przebiegi symulacji gracza wraz z przedziałami ufności.

Źródła losowości są kontrolowane poprzez ustalone ziarna oraz deterministyczną kolejność konfiguracji, a agregacja na poziomie konfiguracji redukuje wariancję wywołaną pojedynczymi losowaniami. Dzięki temu metryki strukturalne i grywalnościowe tworzą spójny zestaw który pozwala jednocześnie porównywać algorytmy oraz analizować wrażliwość na parametry bez ryzyka artefaktów wynikających z jednorazowej lokalizacji *(S/K/E)*.

Eksport danych

Wyniki eksperymentów są eksportowane w formacie `JSON` jako agregaty *konfiguracji* dla każdego algorytmu z osobna. Każdy rekord w pliku odpowiada jednej konfiguracji (tj. konkretnemu wektorowi parametrów) i zawiera opis konfiguracji, rozmiar próby i zagregowane statystyki metryk.

Zmienne, kontrola i plan analizy

Każda konfiguracja, rozumiana jako uporządkowany zestaw parametrów sterujących generacją, wpływa na rozkład własności topologicznych i potencjalny przebieg rozgrywki. Porównania prowadzone są między konfiguracjami w obrębie danego algorytmu, przy stałej procedurze oceny. Dla obliczonych metryk obejmujący wskaźniki strukturalne, grywalnościowe oraz pochodzące z symulacji zachowania gracza raportowana jest wartość średnia, miary rozproszenia oraz odpowiednie przedziały ufności. Takie ujęcie umożliwia zarówno porównania przekrojowe, jak i badania wrażliwości na parametry.

Zakłócenia badania i sposób ich kontroli

W pierwszej kolejności zakłócenia dotyczą losowego rozmieszczenia punktów specjalnych *(S/K/E)*. W celu ograniczenia ich wpływu stosuje się wielokrotne powtórzenia z przetasowaniem położeń oraz agregację wyników na poziomie konfiguracji. Drugim istotnym źródłem zmienności są generator liczby pseudolosowych i jego ziarna. Wartości te są jawnie ustalane i rejestrowane w metadanych, dzięki czemu możliwa jest ścisła replikacja. Trzecim czynnikiem jest wymiar siatki: został on ustalony na stałe, aby nie stanowił odrębnej zmiennej w całym procesie generowania poziomów. Zastosowanie wartości 50×50 wynika z kompromisu pomiędzy wystarczającą rozdzielczością przestrzenną a efektywnością obliczeniową — przy tym rozmiarze można uchwycić istotne zjawiska emergentne bez nadmiernego zwiększania złożoności obliczeniowej. Stała wartość zapewnia porównywalność pomiędzy konfiguracjami, a jednocześnie minimalizuje ryzyko artefaktów skalowych.

Walidacja i wiarygodność wyników

W celu zapewnienia rzetelności i trafności wnioskowań w środowisku badawczym wdrożono wieloetapowe procedury zapewniania jakości danych. Po pierwsze, każda instancja poziomu przechodzi przez test minimalnej grywalności, który eliminuje przypadki niegrywalne i zapewnia, że dalsze analizy dotyczą tylko sensownych konfiguracji. Po drugie, metryki strukturalne i grywalnościowe są obliczane niezależnie, co pozwala na oddzielną ocenę wpływu geometrii od wpływu rozmieszczenia punktów specjalnych. Po trzecie, wyniki są agregowane na poziomie konfiguracji z uwzględnieniem przedziałów ufności, co redukuje wpływ losowości i umożliwia ocenę stabilności wskaźników. Dodatkowo, zastosowano kontrolę źródeł losowości poprzez ustalone ziarna oraz deterministyczną kolejność uruchamiania konfiguracji, co sprzyja replikowalności eksperymentów. Wreszcie, analiza wrażliwości na parametry pozwala na identyfikację potencjalnych artefaktów i zapewnia głębsze zrozumienie dynamiki generowanych poziomów.

Zagrożenia dla trafności i ograniczenia

Wskaźniki uzyskane w symulacjach odnoszą się do zachowania modelowego agenta i nie obejmują pełnego spektrum strategii ludzkich. Ich znaczenie polega przede wszystkim na umożliwieniu porównawczej oceny różnych konfiguracji, a nie na dostarczaniu absolutnej miary jakości rozgrywki. Należy przy tym uwzględnić, że próby generowane w ramach jednej konfiguracji mogą być ze sobą współzależne. Z tego względu podstawową jednostką odniesienia w analizie i raportowaniu jest konfiguracja, a konstrukcja przedziałów ufności została dostosowana do tej struktury danych w taki sposób, aby odzwierciedlać wariancję wewnątrz każdej konfiguracji. Istotnym ograniczeniem jest stałe ustalenie wymiaru planszy, mogące wpływać na rezultaty oraz generować systematyczną stronniczość (bias), którą należy wziąć pod uwagę w interpretacji wyników.

Przedstawione mechanizmy kontroli, rozdzielenie klas metryk oraz analizy czułości służą ograniczeniu wymienionych zagrożeń i podniesieniu wiarygodności uzyskiwanych wyników.

3.2 Symulacja gracza

W celu oceny grywalności wygenerowanych poziomów zastosowano autorską implementację wirtualnego gracza. Symulacja ta stanowi drugą, dynamiczną część procedury badawczej i jest odpowiedzialna za pomiar metryk odzwierciedlających realne doświadczenie gracza w eksploracji mapy. Dzięki temu możliwe jest przełożenie wyników analizy na wnioski dotyczące jakości i ergonomii układu poziomu. Wirtualny gracz został zaprojektowany jako deterministyczny agent sterowany algorytmem eksploracyjno–zadaniowym [24]. Celem jego działania jest odnalezienie klucza, a następnie dotarcie z nim do pola wyjścia. Agent działa w oparciu o cztery podstawowe komponenty funkcjonalne:

Percepcja i model otoczenia. Percepcja realizowana jest poprzez obliczanie pola widzenia (ang. *field of view, FOV*) metodą symetrycznego rzucania cienia (ang. *shadowcasting*) w ośmiu kierunkach. Każde pole, które znalazło się w linii widoczności, jest zapisywane w mapie pól odkrytych, co pozwala agentowi stopniowo budować wewnętrzną reprezentację eksplorowanego obszaru. Mechanizm ten pozwala również na automatyczne dodawanie nowych punktów granicznych (ang. *frontier*) czyli dostępnych pól przylegających do obszarów nieodkrytych, stanowiących potencjalne cele eksploracji. W przypadku zauważenia klucza lub wyjścia ich pozycje są zapisywane w pamięci agenta.

Planowanie ruchu. Poruszanie się w kierunku celu (klucza, wyjścia lub wybranego punktu granicznego) realizowane jest za pomocą zoptymalizowanego algorytmu A* z heurystyką Manhattan [1]. W przypadku braku możliwości wyznaczenia ścieżki w ustalonym

limicie ekspansji, agent stosuje strategię awaryjną polegającą na wyborze ruchu minimalizującego dystans do celu, przy jednoczesnym preferowaniu pól odwiedzanych rzadziej. Logika wyboru celu oparta jest na następującej hierarchii priorytetów:

- Jeśli znana jest pozycja klucza i agent go nie posiada – dążenie do klucza.
- Jeśli agent posiada klucz i zna pozycję wyjścia – dążenie do wyjścia.
- W przeciwnym razie – dążenie do najbliższego punktu granicznego.

Tak zaprojektowana strategia pozwala agentowi efektywnie łączyć eksplorację ukierunkowaną z eksploracją oportunistyczną, ograniczając zjawisko zapętlania i nieproduktywnego ruchu.

Zarządzanie stanem i warunki zakończenia symulacji. Stan wewnętrzny agenta obejmuje bieżącą pozycję, status posiadania klucza, liczniki odwiedzin pól, kolejkę ostatnich ruchów oraz listę znanych celów. Dodatkowo monitorowana jest liczba kolejnych nieudanych ruchów, a przekroczenie ustalonego progu powoduje przedwczesne zakończenie przebiegu. Symulacja kończy się również w przypadku osiągnięcia limitu kroków lub spełnienia warunków sukcesu.

3.3 Opis implementacji badanych algorytmów

W celu przeprowadzenia badań eksperymentalnych konieczne było zaimplementowanie zestawu algorytmów generacji lochów oraz określenie dla nich odpowiednich parametrów sterujących. Parametry te zostały dobrane tak, aby umożliwiały uzyskanie szerokiego spektrum rezultatów – od struktur prostych i ubogich w szczegóły, po złożone i rozbudowane układy przestrzenne.

Przy wyborze zakresów parametrów szczególną uwagę zwrócono na dwa aspekty. Po pierwsze, istotne było uzyskanie zróżnicowanych efektów wizualnych i topologicznych, pozwalających na analizę wpływu poszczególnych metod na strukturę mapy. Po drugie, należało uwzględnić ograniczenia związane z dostępną mocą obliczeniową, tak aby możliwe było przeprowadzenie pełnych serii eksperymentów w rozsądnym czasie.

3.3.1 Implementacja algorytmu *Pijany Spacer* – DrunkardWalk

Proces generacji odbywa się na dwuwymiarowej siatce, w której każda komórka może przyjąć status pola dostępnego lub ściany. Struktura lochu powstaje w wyniku sekwencyjnych ruchów wędrowców, których trajektorie determinują układ korytarzy.

Parametry sterujące

Działanie algorytmu jest regulowane przez zestaw czterech parametrów, pozwalających modelować zarówno stopień chaotyczności, jak i spójność powstających struktur:

- **liczba_kroków** $\in \{400, 800, 1200, 1600\}$ – całkowita liczba wykonanych ruchów; wyższe wartości prowadzą do powstania większej liczby połączeń i rozleglejszej siatki korytarzy,
- **liczba_wędrowców** $\in \{1, 2, 3\}$ – liczba aktywnych agentów; pojedynczy wędrowiec generuje jedną, spójną ścieżkę, podczas gdy wielu wędrowców może wytworzyć bardziej złożoną i połączoną strukturę,
- **prawdopodobieństwo_kroku_wstecz** $\in \{0.0, 0.25, 0.5\}$ – określa szansę wykonania ruchu odwrotnego względem poprzedniego, co sprzyja zagęszczaniu lokalnych fragmentów mapy,
- **preferencja_kierunkowa** $\in \{0.0, 0.3, 0.6\}$ – definiuje tendencję do kontynuowania ruchu w tym samym kierunku; niskie wartości zwiększają chaotyczność trajektorii, wysokie zaś sprzyjają formowaniu bardziej liniowych korytarzy.

Przebieg generacji

1. **Inicjalizacja:**
   - Tworzona jest pusta siatka o zadanym rozmiarze.
   - W centralnym punkcie planszy umieszcza się `liczba_wędrowców` wędrowców.
   - Dla każdego wędrowca początkowy kierunek ruchu ustawiany jest jako niezdefiniowany.
   - Komórki startowe oznaczane są jako podłogę.
2. **Iteracja:** wykonywana jest łącznie `liczba_kroków` ruchów, rozdzielonych pomiędzy wszystkich wędrowców.
   - Dla każdego wędrowca losowany jest kierunek ruchu:
     - z prawdopodobieństwem `prawdopodobieństwo_kroku_wstecz` wybierany jest kierunek przeciwny do poprzedniego,
     - albo z prawdopodobieństwem `preferencja_kierunkowa` wybierany jest ten sam kierunek co poprzednio,
     - albo wybierany jest losowo jeden z pozostałych kierunków.
   - Pozycja wędrowca jest aktualizowana zgodnie z wylosowanym kierunkiem, przy czym współrzędne ograniczane są do granic planszy.
   - Odwiedzona komórka zostaje oznaczona jako podłoga.

Charakterystyka rozwiązania

Opracowana implementacja umożliwia szybkie generowanie nieregularnych układów korytarzy, przypominających naturalne systemy jaskiniowe. Dzięki prostocie i niskim wymaganiom obliczeniowym algorytm ten dobrze sprawdza się jako metoda bazowa w badaniach nad proceduralną generacją poziomów. Zróżnicowane wartości parametrów pozwalają uzyskać szerokie spektrum wyników, od wąskich, tunelowych przejść po rozległe i wielokrotnie połączone sieci korytarzy.

3.3.2 Implementacja algorytmu *Binarnego Podziału Przestrzeni* – BSP

Proces generacji opiera się na rekurencyjnym podziale prostokątnej planszy na coraz mniejsze regiony, w których następnie losowo umieszczane są komnaty. Ostateczna mapa powstaje poprzez wykuwanie wnętrz tych komnat oraz łączenie ich korytarzami w kształcie litery „L".

Parametry sterujące

Implementacja umożliwia kontrolę przebiegu podziału oraz wielkości pomieszczeń za pomocą następujących parametrów:

- **maksymalna_głębokość_podziału** $\in \{4, 5, 6, 7\}$ – maksymalna liczba poziomów rekurencji; większe wartości prowadzą do powstania większej liczby mniejszych pomieszczeń,
- **minimalny_rozmiar_komnaty** $\in \{3, 4, 5\}$ – dolna granica szerokości i wysokości pojedynczego pokoju,
- **minimalny_wymiar_regionu_do_podziału** $\in \{8, 10, 12\}$ – minimalna wielkość regionu, która jeszcze może być dzielona; zapobiega nadmiernemu rozdrobnieniu przestrzeni,
- **margines_podziału** $\in \{2, 3\}$ – minimalna odległość linii cięcia od granic regionu, zapewniająca regularność podziału,
- **współczynnik_skalowania_komnaty** $\in \{0.6, 0.7, 0.8, 0.9\}$ – określa maksymalny rozmiar komnaty względem dostępnego regionu liścia; mniejsze wartości pozostawiają więcej miejsca na korytarze, większe prowadzą do większych i bardziej wypełnionych pokoi.

Przebieg generacji

1. **Inicjalizacja:** definiowany jest początkowy obszar roboczy odpowiadający całej planszy.

2. **Rekurencyjny podział:**

   - Jeżeli bieżący poziom głębokości osiągnął wartość `maksymalna_głębokość_podziału`, dalszy podział nie jest wykonywany.
   - Jeżeli wymiary regionu są mniejsze niż `minimalny_wymiar_regionu_do_podziału` lub linia cięcia nie może zostać poprowadzona z zachowaniem `margines_podziału`, podział zostaje przerwany.
   - W przeciwnym razie region jest dzielony na dwa podregiony, zwykle wzdłuż dłuższego wymiaru; przy zbliżonych proporcjach kierunek cięcia wybierany jest losowo.
   - Podział powtarzany jest rekurencyjnie dla każdego podregionu, zwiększając poziom głębokości.

3. **Generowanie komnat:**

   - W każdym regionie będącym liściem tworzona jest jedna komnata.
   - Wymiary komnaty losowane są tak, aby spełniały warunek `minimalny_rozmiar_komnaty`.
   - Maksymalne wymiary ograniczane są przez `współczynnik_skalowania_komnaty` względem wielkości regionu liścia.
   - Pozycja komnaty ustalana jest losowo w taki sposób, by cała mieściła się w danym regionie.

4. **Łączenie komnat:** wszystkie komnaty w liściach danego poziomu rekurencji są łączone korytarzami w kształcie litery „L”, prowadzonymi pomiędzy ich środkami geometrycznymi. Dzięki pozostawionym marginesom regiony nie nakładają się i korytarze mają miejsce na przebieg.

Charakterystyka rozwiązania

Zaimplementowana wersja algorytmu *BSP* pozwala generować poziomy o przejrzystym i uporządkowanym układzie typu „komnaty–korytarze”. Dzięki parametryzacji można kontrolować zarówno gęstość zabudowy, jak i proporcje pomieszczeń do korytarzy. Algorytm charakteryzuje się deterministyczną strukturą podziału, która gwarantuje brak nakładania się pomieszczeń i pełną spójność mapy, przy jednoczesnym zachowaniu pewnej losowości w rozmieszczeniu i rozmiarach komnat.

3.3.3 Implementacja algorytmu *Automatów Komórkowych* – CellularAutomata

Metoda ta generuje mapę poprzez iteracyjne przekształcanie losowo zainicjalizowanej siatki komórek zgodnie z prostymi regułami progowymi zależnymi od sąsiedztwa. W efekcie z początkowego szumu losowego wyłania się spójna, organiczna struktura przypominająca naturalne jaskinie.

Parametry sterujące

Działanie algorytmu kontrolowane jest przez zestaw czterech parametrów:

- **prawdopodobieństwo_ściany** $\in \{0.40, 0.45, 0.50, 0.55\}$ – określa początkowe zagęszczenie ścian w losowej inicjalizacji; wyższe wartości prowadzą do bardziej zwartej mapy,
- **liczba_iteracji** $\in \{3, 5, 7, 10\}$ – liczba cykli zastosowania reguł automatu; większe wartości wygładzają mapę i prowadzą do powstania większych, spójnych komór,
- **próg_ściany** $\in \{3, 4, 5\}$ – minimalna liczba sąsiadów–ścian w sąsiedztwie Moore'a (8 kierunków), powyżej której dana komórka staje się ścianą,
- **próg_podłogi** $\in \{2, 3, 4\}$ – maksymalna liczba sąsiadów–ścian, przy której komórka staje się podłogą.

Przebieg generacji

1. **Inicjalizacja:**
   - Każda komórka wewnętrzna przyjmuje stan *ściana* z prawdopodobieństwem `prawdopodobieństwo_ściany`, w przeciwnym razie stan *podłoga*.
   - Wszystkie komórki na krawędziach ustawiane są na *ściana*, aby zagwarantować zamknięcie mapy.
2. **Iteracyjne przetwarzanie `liczba_iteracji` razy:**
   - Dla każdej komórki liczone jest, ilu z jej 8 sąsiadów (sąsiedztwo Moore'a) ma stan *ściana*.
   - Następnie stosowane są reguły:
     - jeśli liczba sąsiadów $\geq$ `próg_ściany` – komórka staje się *ściana*,
     - jeśli liczba sąsiadów $\leq$ `próg_podłogi` – komórka staje się *podłoga*,
     - w przeciwnym wypadku komórka zachowuje swój poprzedni stan.
   - Wszystkie zmiany przechowywane są w buforze pomocniczym i wprowadzane jednocześnie po pełnym przejściu przez całą siatkę.

Charakterystyka rozwiązania

Zaimplementowany wariant automatu komórkowego pozwala w prosty sposób generować nieregularne, organiczne układy korytarzy i komór przypominające naturalne jaskinie. Dzięki parametryzacji możliwe jest sterowanie stopniem „szorstkości" struktury: od silnie zaszumionych, fragmentarycznych układów po gładkie i zwarte przestrzenie. Algorytm ten cechuje się niską złożonością obliczeniową i dużą elastycznością, co czyni go popularnym narzędziem w proceduralnej generacji poziomów.

3.3.4 Implementacja algorytmu *Szum Perlina* – PerlinNoise

Procedura generacji polega na próbkowaniu funkcji szumu dla każdej komórki planszy oraz progowaniu uzyskanych wartości. Komórki o wartości powyżej zadanego progu interpretowane są jako pola przechodnie, pozostałe jako ściany.

Parametry sterujące

Algorytm wykorzystuje cztery główne parametry, które pozwalają kształtować charakter wygenerowanych struktur:

- **próg_przepuszczalności** $\in \{0.40, 0.50, 0.60\}$ – wartość graniczna decydująca o tym, które komórki staną się przejściami, niższe wartości prowadzą do większych otwartych przestrzeni,
- **skala** $\in \{12, 16, 24, 48\}$ – kontroluje wielkość globalnych struktur, mniejsze wartości generują duże, rozległe obszary, natomiast większe sprzyjają powstawaniu drobniejszych szczegółów,
- **częstotliwość** $\in \{0.5, 1.0, 1.5, 3.0\}$ – określa stopień zróżnicowania przestrzeni, niskie wartości dają gładkie kształty, wyższe zwiększają liczbę detali,
- **amplituda** $\in \{0.8, 1.0, 1.5\}$ – współczynnik skalujący wartości szumu, umożliwiający wzmocnienie kontrastu między ścianami a przejściami.

Przebieg generacji

1. **Inicjalizacja:**
   - Tworzona jest siatka o zadanym rozmiarze, domyślnie wypełniona ścianami.
   - Na podstawie ziarna losowane są przesunięcia $(\text{offset}_x, \text{offset}_y)$, aby uniknąć powtarzalnych artefaktów szumu.
2. **Próbkowanie:**

- Dla każdej komórki (x, y) obliczane są współrzędne:

$$p_x = \frac{x + \text{offset}_x}{\texttt{skala}} \cdot \texttt{częstotliwość}, \tag{3.1}$$

$$p_y = \frac{y + \text{offset}_y}{\texttt{skala}} \cdot \texttt{częstotliwość}. \tag{3.2}$$

- Dla (p_x, p_y) wyznaczana jest wartość $n = \texttt{PerlinNoise}(p_x, p_y) \cdot \texttt{amplituda}$.

3. **Progowanie:**

- Jeśli $n \geq \texttt{próg\_przepuszczalności}$, komórka zostaje oznaczona jako *podłoga*.
- W przeciwnym przypadku pozostaje jako *ściana*.

Charakterystyka rozwiązania

Algorytm Szumów Perlina pozwala na generowanie płynnych, organicznych struktur o dużej różnorodności wizualnej. Dzięki niewielkiej liczbie parametrów jest prosty w strojenie i zapewnia powtarzalność wyników przy użyciu tego samego ziarna losowego. W porównaniu z *Pijanym Spacerem* oraz *Automatem Komórkowym*, metoda ta daje bardziej harmonijne kształty o łagodnych krawędziach, co czyni ją szczególnie użyteczną przy symulowaniu naturalnych jaskiń oraz terenów.

3.3.5 Implementacja algorytmu *Agregacji Ograniczonej Dyfuzją* – DLA

Generacja poziomu odbywa się poprzez sukcesywne przyłączanie wędrowców do rosnącego klastra startującego od centralnego zarodka. W rezultacie powstaje silnie rozgałęziona, dendrytyczna struktura korytarzy.

Parametry sterujące

Implementacja algorytmu pozwala regulować przebieg procesu agregacji za pomocą następujących parametrów:

- **liczba_wędrowców** $\in \{800, 1600\}$ – liczba cząstek, które mają zostać przyłączone do klastra; określa docelową wielkość mapy,
- **margines_startu** $\in \{4, 6, 8\}$ – odległość od aktualnego promienia klastra, na której losowo rozmieszczani są nowi wędrowcy,
- **margines_usunięcia** $\in \{12, 20, 28\}$ – maksymalna odległość, poza którą wędrowiec zostaje odrzucony, aby uniknąć nadmiernego rozpraszania cząstek,

- **prawdopodobieństwo_przylegania** $\in \{1.0, 0.8\}$ – prawdopodobieństwo, że wędrowiec przyłączy się do klastra po kontakcie; niższe wartości wprowadzają większą losowość i rzadsze połączenia,
- **dryf_do_centrum** $\in \{0.0, 0.25, 0.5\}$ – parametr określający preferencję ruchu w stronę środka planszy; wyższe wartości powodują szybszą agregację i bardziej zwarte klastry.

Przebieg generacji

1. **Inicjalizacja:**
   - Tworzona jest plansza wypełniona ścianami.
   - W centrum umieszczany jest pojedynczy zarodek klastra oznaczony jako podłoga.
2. **Uruchamianie wędrowców:**
   - Nowy wędrowiec startuje na losowej pozycji na okręgu otaczającym aktualny klaster, w odległości `margines_startu` od jego promienia.
3. **Ruch wędrowca:**
   - W każdym kroku wędrowiec wybiera jeden z 8 kierunków.
   - Wybór jest stochastyczny, lecz korygowany przez parametr `dryf_do_centrum`, który zwiększa prawdopodobieństwo ruchu w stronę środka planszy.
   - Jeśli wędrowiec znajdzie się w sąsiedztwie 4-kierunkowym klastra (góra, dół, lewo, prawo), wówczas z prawdopodobieństwem `prawdopodobieństwo_przylegania` staje się częścią klastra (komórka zamieniana na podłogę).
   - Jeżeli wędrowiec oddali się dalej niż `margines_usunięcia` od promienia klastra, zostaje usunięty z symulacji i zastąpiony nowym.
4. **Zakończenie:**
   - Proces powtarzany jest aż do momentu, gdy do klastra przyłączy się `liczba_wędrowców` cząstek.

Charakterystyka rozwiązania

Zaimplementowana wersja DLA umożliwia tworzenie nieregularnych, rozgałęzionych struktur przypominających formacje naturalne (np. korzenie, krystalizacje). Parametryzacja pozwala kształtować zarówno gęstość klastra, jak i stopień jego rozproszenia. Algorytm, mimo swojej prostoty, generuje mapy o wysokiej różnorodności topologicznej.

3.3.6 Implementacja algorytmu *Prosty Pokój* – SimpleRoom

W celu uzyskania punktu odniesienia (ang. *baseline*) zaimplementowano minimalny algorytm generacji *Prosty Pokój* (ang. *Simple Room*). Jego działanie sprowadza się do wykucia pojedynczego kwadratowego pomieszczenia o zadanym boku, umieszczonego w losowym miejscu planszy, a następnie opcjonalnego częściowego wypełnienia go ponownie ścianami. Uzyskana w ten sposób mapa pełni funkcję referencyjną i pozwala ocenić, jak złożoność analizowanych algorytmów przekłada się na metryki strukturalne i grywalnościowe, względem najbardziej trywialnej formy poziomu.

Parametry sterujące

Implementacja przewiduje dwa parametry:

- **długość_boku_kwadratu** $\in \{10, 15, 20, 25, 30\}$ – rozmiar wygenerowanego pomieszczenia; ograniczony do wymiarów planszy,
- **udział_punktów** $\in \{0.0, 0.1, 0.3, 0.4, 0.5\}$ – ułamek pól wewnątrz pokoju, które są losowo przywracane do stanu ściany; umożliwia uzyskanie prostych przeszkód w jednolitej przestrzeni.

Przebieg generacji

1. **Inicjalizacja:**
   - Cała plansza zostaje wypełniona ścianami.
   - Losowo wybierana jest pozycja lewego górnego rogu kwadratu, tak aby cały obszar o boku `długość_boku_kwadratu` mieścił się w granicach planszy.
   - Wszystkie pola w wyznaczonym kwadracie są oznaczane jako *podłoga*.
2. **Wprowadzanie przeszkód:**
   - Spośród pól podłogi wybierana jest losowo część odpowiadająca wartości parametru `udział_punktów`.
   - Wybrane pola są przywracane do stanu *ściana*.
3. **Zakończenie:**
   - Wynikowa mapa zawiera pojedyncze pomieszczenie o boku `długość_boku_kwadratu`, z opcjonalnie rozmieszczonymi przeszkodami wynikającymi z wartości parametru `udział_punktów`.

Charakterystyka rozwiązania

Tak przygotowany algorytm stanowi minimalny przypadek testowy, w którym loch nie zawiera ani korytarzy, ani rozgałęzień. Dzięki temu może pełnić rolę bazy przy porównywaniu bardziej zaawansowanych metod – pozwala sprawdzić, jak metryki reagują na obecność wyłącznie jednego prostego pomieszczenia w zestawieniu z mapami tworzonymi przez inne algorytmy.

3.4 Metryki

Każda metryka będąca wynikiem przeprowadzonych testów nad poziomem jest skalowana do zakresu $[0, 1]$ co umożliwia bezpośrednie porównania pomiędzy algorytmami i konfiguracjami. Niech G będzie prostokątną siatką o rozmiarze $W \times H$, N liczbą wszystkich pól, a $\mathcal{W} \subseteq G$ zbiorem pól podłóg a $|\mathcal{W}|$ oznaczającą ich liczebność. Specjalne węzły to: S (start), K (klucz) i E (wyjście). Do obliczeń ścieżek używany jest zoptymalizowany algorytm A* [1].

3.4.1 Podstawowy test grywalności

Celem testu jest określenie, czy poziom spełnia minimalne warunki grywalności: posiada wystarczającą powierzchnię przechodnią oraz zapewnia istnienie ścieżek między punktami specjalnymi.

Definicja formalna.

$$M_{\text{play}} = \begin{cases} 1, & \dfrac{|\mathcal{W}|}{N} > 0.01 \ \wedge \ (S \leftrightarrow K) \ \wedge \ (K \leftrightarrow E), \\ 0, & \text{w przeciwnym razie.} \end{cases} \tag{3.3}$$

Interpretacja.

- Wartość $M_{\text{play}} = 1$ oznacza, że loch jest minimalnie grywalny: istnieje ścieżka ze startu do wyjścia oraz z klucza do wyjścia, a część przechodnia zajmuje więcej niż 1% planszy.
- Wartość $M_{\text{play}} = 0$ wskazuje na loch niegrywalny (zbyt mały obszar przechodni albo brak wymaganych połączeń).

Znaczenie testu. Test podstawowej grywalności pełni rolę filtra wstępnego: tylko jeśli zostanie zaliczony ($M_{\text{play}} = 1$), wykonywane są pozostałe metryki. W przeciwnym razie układ lochu jest odrzucany jako niegrywalny i dalsze analizy nie są prowadzone.

3.4.2 Procent przechodniej powierzchni

Celem testu jest określenie, jaka część całkowitej powierzchni lochu nadaje się do poruszania się po niej przez postać gracza. Wskaźnik ten wyznacza relację między liczbą pól dostępnych do przejścia a liczbą wszystkich pól w siatce:

$$M_{\text{przech}} = \frac{|\mathcal{W}|}{N} \in [0, 1]. \tag{3.4}$$

W powyższym wzorze $|\mathcal{W}|$ oznacza liczbę pól przechodnich (podłóg), a N jest całkowitą liczbą pól w siatce G.

Interpretacja.

- Wartości M_{przech} bliskie 1 wskazują na układy o dużych, otwartych przestrzeniach, które sprzyjają swobodnemu manewrowaniu natomiast wartości bliskie 0 sugerują ciasne, mało grywalne lochy.

- Wskaźnik jest niezależny od bezwzględnego rozmiaru planszy: porównuje względny udział pól przechodnich, dzięki czemu ułatwia zestawianie różnych układów lochów tej samej lub odmiennej wielkości.

W praktyce metryka ta dostarcza szybkiej, globalnej oceny „otwartości" poziomu i może służyć jako kryterium wstępnej selekcji układów (np. odrzucanie zbyt ciasnych lub zbyt pustych planów).

3.4.3 Stosunek ślepych zaułków

Celem testu jest obliczenie odsetka pól przechodnich, które tworzą ślepe zaułki czyli mają dokładnie jednego sąsiada również należącego do zbioru pól przechodnich.

Formalnie definiujemy zbiór:

$$D = \{\, v \in \mathcal{W} \;:\; \deg_{\mathcal{W}}(v) = 1 \,\}. \tag{3.5}$$

gdzie $\deg_{\mathcal{W}}(v)$ oznacza liczbę sąsiednich pól przechodnich dla pola v. Następnie metryka przyjmuje postać:

$$M_{\text{zaułki}} = \frac{|D|}{|\mathcal{W}|}, \quad M_{\text{zaułki}} \in [0, 1]. \tag{3.6}$$

Wysokie wartości tego wskaźnika wskazują, że układ przypomina labirynt o wielu martwych końcach, co może utrudniać eksplorację i zmuszać gracza do częstych powrotów. Niskie wartości sugerują, że loch jest bardziej „otwarty" i ma mniej pułapkowych struktur.

3.4.4 Stosunek liniowych korytarzy

Celem testu jest określenie, jaki odsetek pól przechodnich tworzy odcinki prostych korytarzy, tzn. takich pól, które mają dokładnie dwóch przeciwległych sąsiadów przechodnich i jednocześnie brak sąsiadów w kierunkach prostopadłych. Podczas sprawdzania sąsiedztwa uwzględnia się jedynie komórki mieszczące się w granicach siatki.

Definicja formalna. Niech $\deg_{\mathcal{W}}(v)$ oznacza liczbę sąsiadów przechodnich pola v. Definiujemy zbiór

$$\mathcal{L} = \{\, v \in \mathcal{W} \;:\; \deg_{\mathcal{W}}(v) = 2 \;\text{ i sąsiedzi } v \text{ leżą naprzeciw siebie} \,\}. \tag{3.7}$$

Wzór.

$$M_{\text{lin}} = \frac{|\mathcal{L}|}{|\mathcal{W}|}, \qquad M_{\text{lin}} \in [0, 1]. \tag{3.8}$$

Interpretacja.

- Wysokie wartości M_{lin} oznaczają układy z przewagą prostych, długich ciągów korytarzy, mniejszą liczbą skrzyżowań i zakrętów sprawia, że rozgrywka bywa bardziej przewidywalna.
- Niskie wartości wskazują na większą różnorodność topologii: częstsze skrzyżowania, zakręty i komnaty, co sprzyja złożonej eksploracji.

3.4.5 Średni stopień węzła

Celem testu jest określenie, jak bardzo „otwarty" jest układ lochu, poprzez zbadanie średniej liczby sąsiadów przechodnich przypadającej na jedno pole. Wartość ta opisuje przeciętną liczbę możliwych kierunków ruchu w dowolnym punkcie planszy.

Definicja formalna. Niech $\deg_{\mathcal{W}}(v)$ oznacza liczbę sąsiadów pola $v \in \mathcal{W}$, którzy również należą do zbioru pól przechodnich. Definiujemy średni stopień:

$$\bar{d} = \frac{1}{|\mathcal{W}|} \sum_{v \in \mathcal{W}} \deg_{\mathcal{W}}(v). \tag{3.9}$$

Ponieważ maksymalny stopień sąsiedztwa w siatce wynosi 8, wprowadzamy znormalizowaną miarę:

$$M_{\text{deg}} = \frac{\bar{d}}{8}, \qquad M_{\text{deg}} \in [0, 1]. \tag{3.10}$$

Interpretacja.

- Wysokie wartości M_{deg} wskazują na plansze bardziej otwarte, o dużej liczbie skrzyżowań i swobodzie wyboru drogi.

- Niskie wartości oznaczają układy bardziej korytarzowe, w których większość pól ma ograniczoną liczbę sąsiadów, co sprzyja powstawaniu tuneli i ślepych zaułków.

3.4.6 Przesunięcie centroidu

Celem testu jest zbadanie, na ile rozkład przestrzeni przechodniej w lochu jest symetryczny względem geometrycznego środka planszy. Wskaźnik porównuje położenie centroidu (środka masy pól przechodnich) z punktem centralnym siatki.

Definicja formalna. Centroid pól przechodnich definiujemy jako:

$$c = \left(\frac{1}{|\mathcal{W}|} \sum_{(x,y)\in\mathcal{W}} x,\ \frac{1}{|\mathcal{W}|} \sum_{(x,y)\in\mathcal{W}} y \right). \tag{3.11}$$

Geometryczny środek planszy o wymiarach $W \times H$ to:

$$g = \left(\tfrac{W-1}{2},\ \tfrac{H-1}{2} \right). \tag{3.12}$$

Metryka przesunięcia centroidu przyjmuje postać:

$$M_{\text{centroid}} = \frac{\|c - g\|_2}{\sqrt{\left(\frac{W-1}{2}\right)^2 + \left(\frac{H-1}{2}\right)^2}}, \qquad M_{\text{centroid}} \in [0, 1]. \tag{3.13}$$

Interpretacja.

- $M_{\text{centroid}} = 0$ oznacza idealnie symetryczne rozłożenie przestrzeni względem środka planszy.

- Wartości bliskie 1 wskazują na silnie przesuniętą, niesymetryczną strukturę, w której pola przechodnie są skupione przy jednej z krawędzi.

3.4.7 Wskaźnik zakrętów na optymalnej ścieżce

Celem testu jest ocena płynności najkrótszej trasy prowadzącej przez punkty specjalne. Miarą jest udział zakrętów w stosunku do maksymalnej możliwej ich liczby.

Definicja formalna. Niech m oznacza długość ścieżki (liczbę krawędzi), a t jako liczbę zakrętów, czyli zmian kierunku między kolejnymi krokami. Definiujemy:

$$M_{\text{turn}} = \begin{cases} 1, & m < 2, \\ 1 - \dfrac{t}{m-1}, & \text{w przeciwnym razie.} \end{cases} \tag{3.14}$$

Interpretacja.

- Wysokie wartości M_{turn} oznaczają, że trasa jest płynna i zbliżona do prostoliniowej.
- Niskie wartości wskazują na krętą ścieżkę wymagającą wielu zmian kierunku.

Uwagi zgodności z implementacją. W implementacji najpierw wyznaczana jest najkrótsza ścieżka z S do K oraz z K do E. Obie są łączone w jedną trasę (z usunięciem duplikatu w punkcie K). Następnie liczona jest liczba zakrętów, tj. miejsc, gdzie wektor kierunku ulega zmianie. Wynik normalizowany jest względem maksymalnej możliwej liczby zakrętów $(m-1)$, a końcowa wartość ograniczana do przedziału $[0, 1]$.

3.4.8 Analiza ścieżki krytycznej

Celem testu jest określenie, jak długi w relacji do całego lochu jest odcinek obowiązkowej trasy $S \to K \to E$, czyli ścieżki krytycznej. Metryka ta odzwierciedla minimalną długość, jaką gracz musi pokonać, aby ukończyć poziom.

Definicja formalna. Niech $d(u, v)$ oznacza długość najkrótszej ścieżki między polami u i v. Krytyczna trasa obejmuje przejście ze startu S do klucza K oraz z klucza do wyjścia E. Aby uniknąć podwójnego zliczenia pola K, odejmujemy 1:

$$M_{\text{crit}} = \frac{d(S, K) + d(K, E) - 1}{N}, \qquad M_{\text{crit}} \in [0, 1]. \tag{3.15}$$

gdzie $N = W \times H$ oznacza liczbę wszystkich pól w siatce.

Interpretacja.

- Niższe wartości M_{crit} oznaczają lochy bardziej kompaktowe, w których droga wymagana do ukończenia jest krótka w stosunku do rozmiaru planszy.
- Wyższe wartości wskazują na rozciągnięte układy, gdzie ścieżka krytyczna obejmuje znaczną część całej planszy.

3.4.9 Test wymaganego cofania

Celem testu jest ocena, o ile dłuższa jest obowiązkowa trasa $S \to K \to E$ względem bezpośredniej drogi $S \to E$. Wysoki wynik oznacza, że gracz musi znacząco nadłożyć drogi, by zdobyć klucz, zanim dotrze do wyjścia.

Definicja formalna. Niech $d(u, v)$ oznacza długość najkrótszej ścieżki pomiędzy punktami u i v. Wówczas:

$$\Delta = (d(S, K) + d(K, E) - 1) - d(S, E), \tag{3.16}$$

$$M_{\text{extra}} = \frac{\Delta}{d(S, E)}. \tag{3.17}$$

Interpretacja.

- $M_{\text{extra}} = 0$ oznacza brak dodatkowego cofania — trasa przez klucz praktycznie nie wydłuża drogi.
- Wysokie wartości M_{extra} wskazują na konieczność znacznego nadłożenia drogi, co może czynić rozgrywkę bardziej uciążliwą.

3.4.10 Zdobyte klucze przez gracza

Celem tego wskaźnika jest określenie, jak często gracz w symulacji odnajduje klucz. Mierzy się udział udanych prób w całkowitej liczbie powtórzeń symulacji.

Definicja formalna. Przyjmijmy n — liczbę niezależnych symulacji. Niech $\mathbf{1}\{\cdot\}$ oznacza funkcję wskaźnikową, przyjmującą wartość 1, jeśli warunek jest spełniony (gracz znalazł klucz), oraz 0 w przeciwnym razie. Definiujemy:

$$M_{\text{key}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{\text{gracz znalazł klucz}\}, \qquad M_{\text{key}} \in [0, 1]. \tag{3.18}$$

Interpretacja.

- $M_{\text{key}} = 1$ oznacza, że we wszystkich przebiegach gracze odnajdowali klucz.
- $M_{\text{key}} = 0$ oznacza, że klucz nigdy nie został odnaleziony.
- Wartości pośrednie wskazują na trudność poziomu: im niższy wskaźnik, tym większe ryzyko, że klucz pozostanie niewykryty podczas gry.

3.4.11 Udane wyjście z poziomu przez gracza

Celem tego wskaźnika jest określenie, jak często gracz w symulacji dociera do wyjścia po uprzednim zdobyciu klucza. Mierzy on odsetek udanych ukończeń poziomu w wielu powtórzeniach rozgrywki.

Definicja formalna. Dla n niezależnych symulacji:

$$M_{\text{exit}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{\text{gracz znalazł klucz i wyszedł z poziomu}\}, \qquad M_{\text{exit}} \in [0, 1]. \tag{3.19}$$

Interpretacja.

- $M_{\text{exit}} = 1$ oznacza, że gracz zawsze kończył poziom.
- $M_{\text{exit}} = 0$ oznacza, że żadna symulacja nie zakończyła się sukcesem.
- Wartości pośrednie wskazują na różny poziom trudności lochu: im niższy wskaźnik, tym większe prawdopodobieństwo porażki gracza.

3.4.12 Procent odkrytego poziomu

Celem tego wskaźnika jest określenie, jaką część pól przechodnich gracz odkrył podczas eksploracji poziomu przez gracza.

Definicja formalna.

$$M_{\text{cover}} = \frac{\text{ilość odkrytych pól}}{|\mathcal{W}|}, \qquad M_{\text{cover}} \in [0, 1]. \tag{3.20}$$

Interpretacja.

- Wartości bliskie 1 oznaczają, że gracz zobaczył niemal cały loch.
- Niskie wartości wskazują, że duża część przestrzeni pozostała nieodkryta.

Uwagi zgodności z implementacją. W implementacji liczba odkrytych pól to suma pól przechodnich, które znalazły się w polu widzenia gracza przynajmniej raz w trakcie symulacji.

3.4.13 Kroki do wyjścia

Metryka ta mierzy liczbę kroków potrzebnych do ukończenia poziomu przez gracza (od startu S do wyjścia E, po zdobyciu klucza K), znormalizowaną względem całkowitej liczby pól planszy.

Definicja formalna.

$$M_{\text{steps}} = \frac{\text{ilość kroków od startu do wyjścia}}{N}, \qquad M_{\text{steps}} \in [0,1]. \tag{3.21}$$

Interpretacja.

- Niskie wartości oznaczają kompaktowe poziomy, gdzie droga do wyjścia jest relatywnie krótka.
- Wysokie wartości sugerują rozległe lochy, w których ukończenie wymaga wielu kroków.

3.4.14 Średnie odkrycie poziomu na krok

Wskaźnik ten mierzy, ile nowych pól przechodnich gracz odkrywa średnio na jeden krok.

Definicja formalna.

$$M_{\text{eff}} = \frac{M_{\text{cover}}}{\text{liczba kroków}}, \qquad M_{\text{eff}} \in [0,1]. \tag{3.22}$$

Interpretacja.

- Wysokie wartości oznaczają, że gracz efektywnie odkrywa kolejne części planszy.
- Niskie wartości wskazują na liczne powroty i marnowanie kroków na już znane obszary.

Uwagi zgodności z implementacją. W implementacji efektywność obliczana jest jako stosunek procentu odkrytych pól do liczby kroków wykonanych przez gracza w danej symulacji - nawet tych zakończonych niepowodzeniem.

3.4.15 Prędkość odkrycia poziomu przez gracza

Celem tej metryki jest ocena tempa odkrywania planszy. Mierzy ona pole pod krzywą wzrostu liczby odkrytych pól w czasie.

Definicja formalna.

$$M_{\text{AUC}} = \frac{\sum_{t=1}^{T} C_t}{T \cdot |\mathcal{W}|}, \qquad M_{\text{AUC}} \in [0,1]. \tag{3.23}$$

gdzie C_t to liczba odkrytych pól przechodnich w kroku t, a T to całkowita liczba kroków w symulacji.

Interpretacja.

- Wysokie wartości oznaczają, że gracz szybko eksploruje duże fragmenty planszy.
- Niskie wartości sugerują powolne odkrywanie i długotrwałe błądzenie.

Uwagi zgodności z implementacją. W implementacji po każdym kroku zapisywana jest liczba pól odkrytych przez gracza. Sumaryczna wartość dzielona jest przez maksymalny możliwy wynik.

3.4.16 Stagnacje gracza

Metryka ta mierzy, jak często gracz wracał na pola odwiedzone wcześniej.

Definicja formalna.

$$M_{\text{revisit}} = 1 - \frac{\text{liczba unikalnie odwiedzonych pól}}{\text{ilość kroków}}, \qquad M_{\text{revisit}} \in [0, 1]. \tag{3.24}$$

Interpretacja.

- $M_{\text{revisit}} = 0$ oznacza, że każdy krok prowadził na nowe pole (brak powrotów).
- Wysokie wartości wskazują na częste powroty, stagnacje lub błądzenie gracza.

Uwagi zgodności z implementacją. W implementacji zliczana jest liczba wszystkich kroków oraz liczba unikalnych pól, na które gracz wszedł. Różnica odzwierciedla liczbę powtórnych odwiedzin. Wynik normalizowany jest względem liczby kroków i ograniczony do $[0, 1]$.

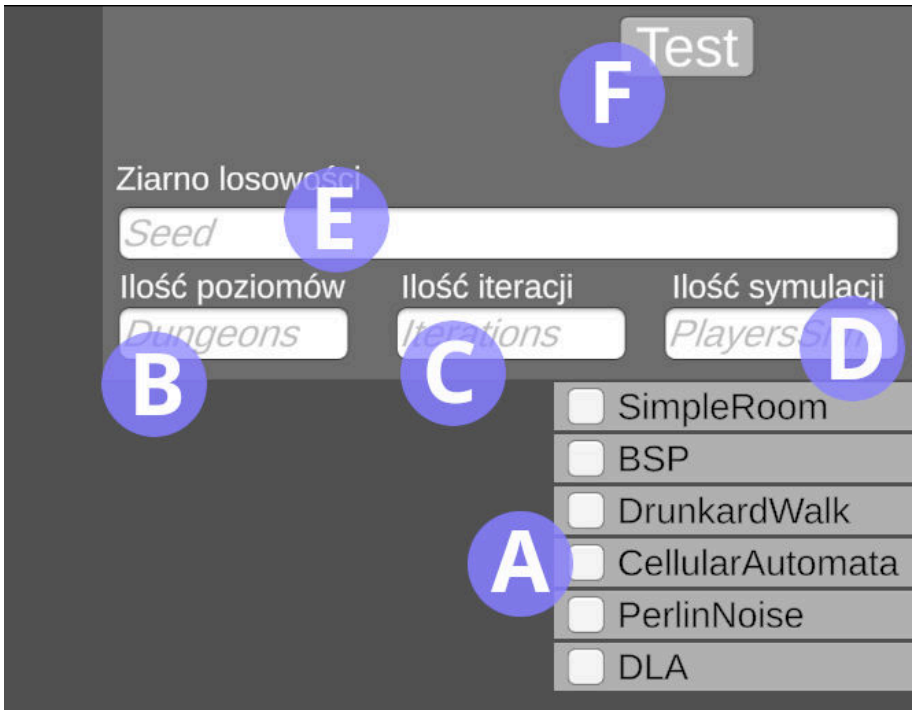
3.5 Przeprowadzenie eksperymentu

W tej sekcji opisano procedurę badawczą z perspektywy użytkownika systemu, jak przygotować konfigurację, jak przebiega eksperyment oraz jakie są artefakty wyjściowe.

Przygotowanie konfiguracji

Eksperyment rozpoczyna się od przygotowania danych wejściowych w interfejsie konfiguracyjnym (rys. 3.1). Badacz:

- (A) wskazuje algorytmy poddawane analizie,
- (B) ustala liczbę lochów generowanych dla każdej konfiguracji,



Rysunek 3.1: Zrzut ekranu przedstawiający interfejs konfiguracyjny przed uruchomieniem eksperymentu.

- (C) określa liczbę iteracji przetasowania punktów specjalnych,
- (D) definiuje liczbę przebiegów symulowanego gracza,
- (E) ustala początkowe ziarno losowości,
- (F) uruchamia proces generowania poziomów.

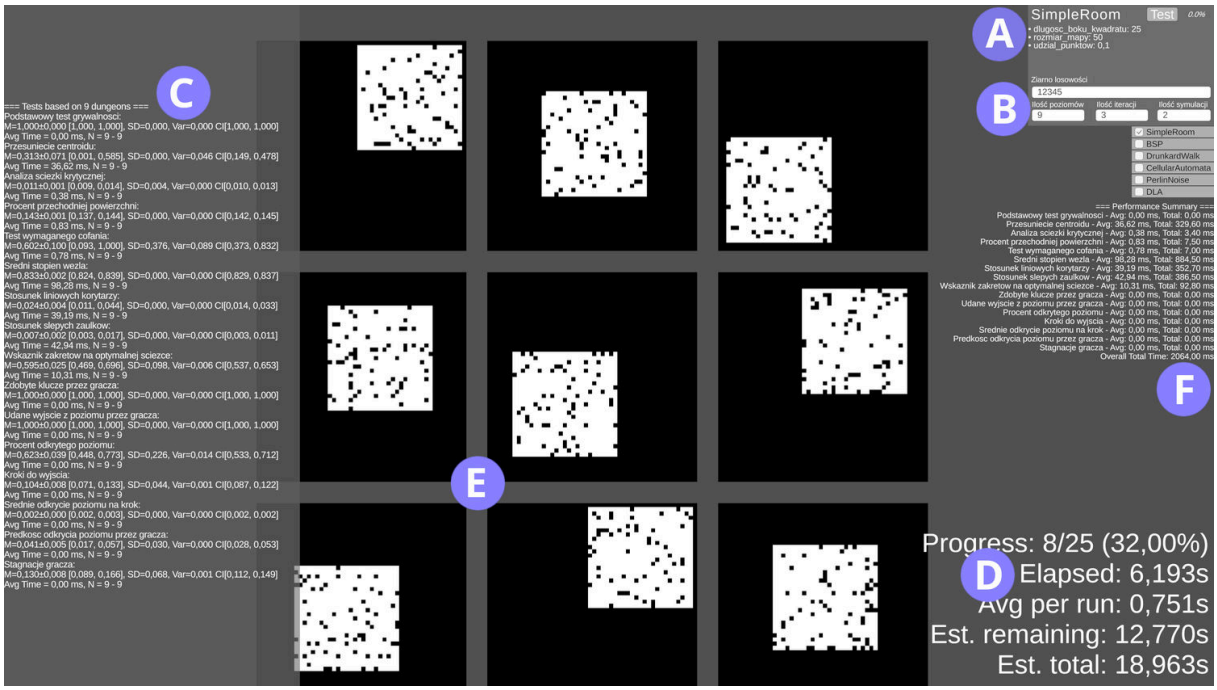
Na podstawie tych ustawień system oblicza całkowitą liczbę konfiguracji do przetestowania, co pozwala oszacować skalę eksperymentu oraz czas jego realizacji.

Uruchomienie i monitoring

Po uruchomieniu generowania system automatycznie wytwarza paczki lochów dla każdej kombinacji parametrów. Jeśli aktywna jest wizualizacja, poziomy są odtwarzane w trójwymiarowej scenie *Unity*, a ich układ może być zapisywany w postaci zrzutów ekranu. Równolegle przebiega weryfikacja minimalnej grywalności, a dla pozytywnie zweryfikowanych poziomów — uruchamiane są testy szczegółowe i wielokrotne symulacje wirtualnego gracza.

Podczas trwania eksperymentu badacz ma bieżący wgląd w postęp (rys. 3.2). Interfejs prezentuje:

- (A) konfigurację aktualnie badanego algorytmu,
- (B) parametry badania,



Rysunek 3.2: Zrzut ekranu przedstawiający przebieg badania i podgląd wyników w czasie rzeczywistym.

- (C) wyniki testów cząstkowych,
- (D) przewidywany czas zakończenia,
- (E) wizualizację wygenerowanych poziomów,
- (F) średnie czasy wykonywania poszczególnych testów.

Zapis i eksport wyników

Po zakończeniu eksperymentu wyniki są zapisywane do plików `JSON`. Zawierają one szczegółowe raporty dla każdej konfiguracji, wartości uśrednione, miary niepewności oraz metadane eksperymentu. Tak przygotowany zestaw wyników stanowi wejście dla warstwy analityczno-wizualnej odpowiedzialnej za obliczenia statystyczne i wizualizację wyników.

3.6 Architektura środowiska badawczego

W tej sekcji przedstawiono strukturę i implementację środowiska, podzielonego na dwie warstwy: *eksperymentalną* (Unity) oraz *analityczno-wizualną* (potok analityczny).

Warstwa eksperymentalna

Warstwa eksperymentalna została zbudowana w środowisku *Unity* jako narzędzie do automatycznego badania algorytmów proceduralnego generowania poziomów. Architek-

tura ma charakter modularny, co rozdziela odpowiedzialności między:

- komponenty sterujące przebiegiem badań,
- moduły generujące poziomy,
- komponenty testowe,
- warstwę wizualną.

Centralną rolę pełni klasa `ResearchPerformer`, odpowiedzialna za koordynację eksperymentu: komunikację z użytkownikiem, gromadzenie parametrów wejściowych, inicjalizację algorytmów, przekazywanie wygenerowanych poziomów do modułu testowego oraz kontrolę zapisu wyników. Jej uzupełnieniem jest `DungeonTestConfiguration` — wyspecjalizowany system analityczny, w którym uruchamiane są testy strukturalne i symulacyjne.

Generatory poziomów implementują interfejs `IDungeon` i tworzą lochy w postaci dwuwymiarowych siatek binarnych. Dla każdej konfiguracji parametrów powstaje paczka poziomów o wielkości zadanej przez użytkownika. W trybie wizualnym poziomy są renderowane w scenie 3D, a ich układy mogą być eksportowane jako zrzuty ekranu. Analiza przebiega wielowątkowo, z równoległym uruchamianiem testów i symulacji, co zwiększa przepustowość obliczeń.

Dane z testów są agregowane w strukturach raportowych. System oblicza średnie metryk, odchylenia standardowe, wariancję, błędy standardowe oraz przedziały ufności. Mechanizmy buforowania w klasie `TestStatistics` ograniczają redundantne przeliczenia i przyspieszają analizę. Końcowe artefakty są zapisywane w formacie `JSON` wraz z metadanymi eksperymentu.

Warstwa analityczno-wizualna

Warstwa analityczno-wizualna realizuje deterministyczny, replikowalny potok przetwarzania danych. Na wejściu znajdują się pliki wynikowe poszczególnych algorytmów, które są wczytywane i weryfikowane pod kątem kompletności. Etykiety konfiguracji są automatycznie dekomponowane do zmiennych tabelarycznych, co umożliwia bezpośrednie filtrowanie i porównywanie parametrów bez dodatkowych przekształceń. Po normalizacji obserwacje są scalane do jednej, kanonicznej tabeli pełniącej źródło prawdy dla kolejnych etapów.

Na tej podstawie wyznaczane są syntetyczne podsumowania: średnie wartości metryk, miary rozrzutu oraz przedziały ufności, które kwantyfikują niepewność estymacji i pozwalają oceniać istotność różnic. Analizowane są również zależności: wpływ parametrów na jakość przez korelacje rang Spearmana oraz współzmienność metryk przez korelacje Pearsona, co pozwala identyfikować miary o zbliżonej charakterystyce.

Potok generuje tabele i zestaw rysunków: porównania algorytmów dla poszczególnych metryk, wygładzone rozkłady, przekroje wpływu parametrów oraz mapy ciepła zależności między metrykami. Te same dane wejściowe deterministycznie prowadzą do tych samych tabel i rysunków, co zapewnia odtwarzalność. Dodanie nowej metryki, parametru lub algorytmu nie wymaga zmiany logiki potoku — wystarczy dostarczyć odpowiednie pliki wejściowe, a mechanizmy normalizacji, scalania i analizy zastosują się automatycznie.

Rozdział 4

Badania

4.1 Wprowadzenie do badań

Celem niniejszych badań jest przeprowadzenie ilościowej analizy oraz porównania wybranych algorytmów PCG pod kątem ich cech strukturalnych oraz właściwości wpływających na grywalność. Analiza opiera się na zestawie metryk obliczanych automatycznie na podstawie wygenerowanych poziomów. Uwzględniono zarówno metryki strukturalne, opisujące topologiczne właściwości układu poziomów, jak i metryki związane z grywalnością, wyznaczane na podstawie symulowanego zachowania wirtualnego gracza poruszającego się po wygenerowanym poziomie. Uzyskane wyniki mają umożliwić głębsze zrozumienie charakterystyki poszczególnych algorytmów oraz zidentyfikowanie zależności pomiędzy wartościami parametrów wejściowych a właściwościami końcowego poziomu. Zakres badań ograniczono do algorytmów operujących na dwuwymiarowych siatkach, w których poszczególne pola przyjmują wartości binarne, reprezentując obszary dostępne i niedostępne dla gracza. Wyłączono z analizy zarówno algorytmy generujące struktury trójwymiarowe, jak i te wymagające dodatkowych danych wejściowych poza zestawem zdefiniowanych parametrów. Badania przeprowadzono w szerokim zakresie konfiguracji parametrów wejściowych dla każdego algorytmu, co pozwoliło uchwycić wpływ zmian tych parametrów na jakość i charakter wygenerowanych poziomów.

Praca ma charakter przeglądowo-analityczny, a prowadzone badania nie mają na celu wyłonienia jednego, najlepszego algorytmu. Ich zadaniem jest dostarczenie możliwie obiektywnego opisu każdego z analizowanych rozwiązań, pozwalającego zarówno na zrozumienie ich charakterystyki, jak i na określenie wpływu parametrów wejściowych na ostateczny kształt generowanego poziomu. W pracy nie formułuje się hipotez w tradycyjnym ujęciu naukowym, lecz koncentruje się na jak najbardziej pełnym i bezstronnym przedstawieniu badanych metod w określonych warunkach symulacyjnych. Wyniki zaprezentowane są w sposób umożliwiający porównanie poszczególnych rozwiązań, jednak bez wskazywania jednego zwycięzcy. Podejście to pozwala na identyfikację mocnych i słabych stron każdego

algorytmu oraz określenie kontekstu, w którym może on znaleźć szczególne zastosowanie.

Uzasadnienie wyboru algorytmów

Dobór algorytmów poddanych analizie został przeprowadzony z uwzględnieniem szeregu kryteriów, które miały na celu zapewnienie możliwie pełnego i reprezentatywnego obrazu badanego zagadnienia. Podstawowymi wyznacznikami wyboru była ich powszechność oraz praktyczne zastosowanie w obszarze projektowania i tworzenia gier komputerowych, a także udokumentowana obecność w literaturze technicznej oraz naukowej, co stanowiło gwarancję ich rozpoznawalności i znaczenia w kontekście badań nad generacją proceduralną. Równocześnie zwracano uwagę na to, aby analizowany zestaw obejmował algorytmy reprezentujące zróżnicowane podejścia koncepcyjne, oparte na odmiennych zasadach działania, co umożliwia uchwycenie szerokiego spektrum metod i strategii stosowanych w tym obszarze. Dodatkowym, nie mniej istotnym kryterium, była dostępność stabilnych implementacji poszczególnych algorytmów bądź możliwość ich opracowania w sposób zapewniający rzetelne przeprowadzenie testów w spójnym i jednolitym środowisku badawczym. Dzięki takiemu podejściu możliwe stało się przygotowanie zestawu narzędzi analitycznych, który pozwala nie tylko na dokonanie porównań pomiędzy poszczególnymi rozwiązaniami, lecz także na szczegółowe rozpoznanie różnorodności efektów generacji. W konsekwencji uzyskano podstawę do rzetelnej oceny tego, w jaki sposób różne strategie i mechanizmy implementacyjne przekładają się na właściwości generowanych poziomów.

Metodologia badań

Proces badawczy oparto na autorskim systemie umożliwiającym zautomatyzowane przeprowadzanie testów oraz gromadzenie wyników w postaci ujednoliconej bazy danych. Dla każdego algorytmu zdefiniowano zestaw parametrów wejściowych oraz zakresy ich wartości. Następnie, poprzez zastosowanie iloczynu kartezjańskiego tych zakresów, wygenerowano komplet konfiguracji testowych, który dla większości algorytmów liczył kilkaset odmiennych wariantów. Każdy poziom tworzony był w oparciu o deterministyczny generator liczb losowych, co zapewniało pełną powtarzalność wyników. Dla każdej konfiguracji algorytmu generowano zbiór dwa tysiące poziomów (wartość ta, podobnie jak pozostałe parametry eksperymentu, została dostosowana do dostępnych zasobów obliczeniowych). Następnie, dla każdego z wygenerowanych lochów, przeprowadzano proces dziesięciokrotnego losowania położeń punktów specjalnych, takich jak pole startowe, pole z kluczem oraz pole z wyjściem, za każdym razem uzyskując inny układ specjalnych węzłów w obrębie tej samej struktury przestrzennej. Na każdym z tak przygotowanych wariantów mapy uruchamiano trzykrotnie zaawansowaną symulację wirtualnego gracza, wykorzystując algorytm sterowania zapewniający spójne zachowanie w identycznych warunkach wejściowych. Dzięki temu możliwe było uśrednienie wyników zarówno w obrębie pojedyn-

czego rozlosowania pozycji węzłów, jak i w obrębie całej grupy poziomów powiązanych z daną konfiguracją algorytmu. Tak rozbudowana procedura testowa pozwalała na istotne zwiększenie wiarygodności uzyskanych wartości średnich, ograniczając wpływ pojedynczych odchyleń i przypadków skrajnych.

Badania obejmowały dwa zasadnicze etapy. Pierwszy z nich polegał na analizie struktury wygenerowanych map przy użyciu zestawu metryk opisujących ich właściwości topologiczne, drugi etap polegał na uruchomieniu symulacji wirtualnego gracza, której celem była ocena grywalności wygenerowanego poziomu. Każda konfiguracja może być odtworzona w przyszłości, a wyniki weryfikowane lub rozszerzane o dodatkowe testy. Tak szczegółowy i wieloetapowy sposób gromadzenia danych pozwalał na uzyskanie wyników o wysokiej wiarygodności statystycznej, co jest kluczowe przy porównywaniu zróżnicowanych algorytmów generacji proceduralnej.

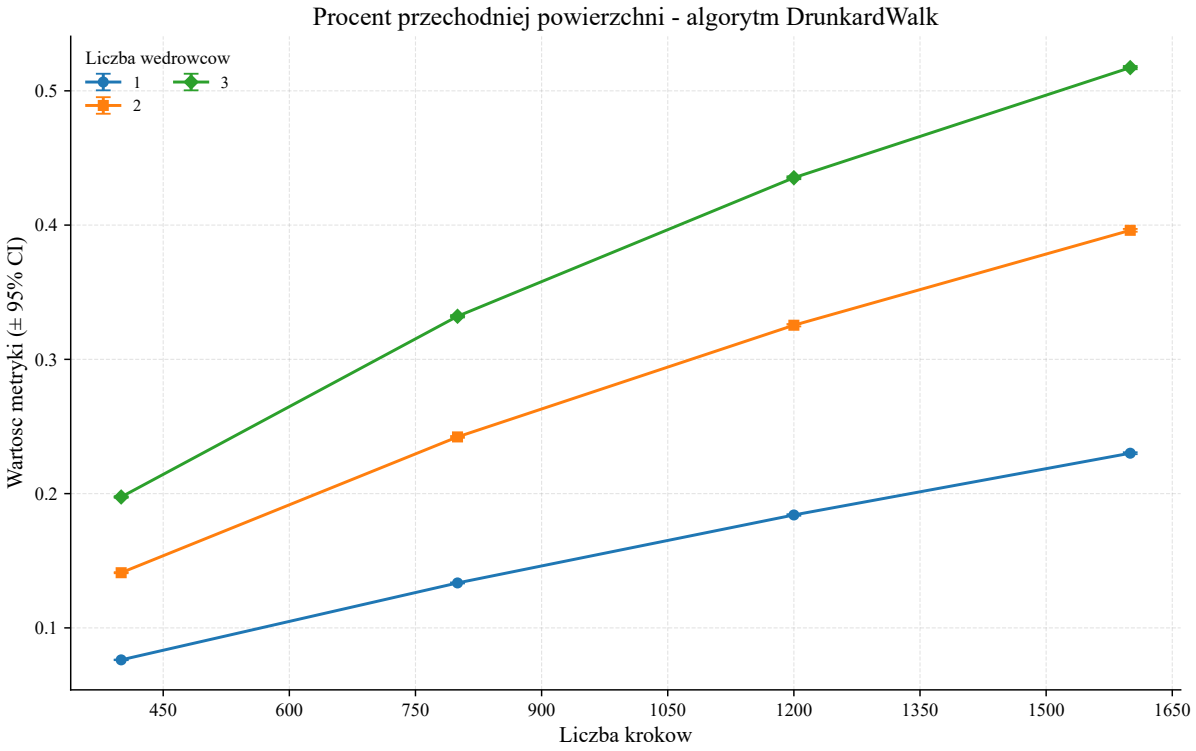
4.2 Algorytm *Pijany Spacer*, DrunkardWalk – studium przypadku

Analiza ilościowa wpływu czterech parametrów algorytmu *Pijany Spacer*: *liczba kroków*, *liczby wędrowców*, *prawdopodobieństwo kroku wstecz* oraz *preferencja kierunkowa* na zestaw metryk topologicznych i grywalnościowych poziomu. Wyniki poddano również analizie korelacji Spearmana, a istotne wartości uwzględniono przy poziomie $p < 0{,}05$. Łącznie przeanalizowano 108 konfiguracji, co odpowiada około $6{,}4 \times 10^6$ pojedynczym przebiegom symulacji gracza. Plik *data_DrunkardWalk.json* z wynikami eksperymentów dołączono do pracy w załączniku.

Powierzchnia przechodnia a liczba wędrowców i kroków.

Dla jednego wędrowcy wraz ze wzrostem liczby kroków sieć korytarzy wyraźnie gęstnieje: pojedyncze ścieżki zaczynają się łączyć, a mapa coraz mniej przypomina zestaw izolowanych odcinków. Przekłada się to na stabilny przyrost powierzchni możliwej do przejścia przy wartościach rzędu $\sim$ 0,08 przy 400 krokach do około $\sim$ 0,23 przy 1600 kroków, każdy kolejny wędrowiec zwiększa ten efekt, lecz w coraz mniejszym stopniu ze względu na nachodzące na siebie ścieżki (rys. 4.1). Jednocześnie układ staje się bardziej skomplikowany: ubywa długich, liniowych przejść i ślepych zakończeń, a optymalna trasa do wyjścia prowadzi częściej przez łączenia i łuki niż przez proste odcinki. Efekt widać w relatywnej długości tej trasy, która rośnie z $\sim$ 0,07 do $\sim$ 0,19 (rys. 4.4).

Każdy kolejny wędrowiec wzmacnia wszystkie powyższe tendencje. Front eksploracji rozsuwa się szerzej, dzięki czemu rośnie zarówno zasięg, jak i łączność między fragmentami siatki. Dla górnego progu kroków powierzchnia przechodnia zbliża się do $\sim$ 0,40, a odsetek ślepych zaułków jest jeszcze mniejszy niż w wariancie z jednym wędrowcem (rys.4.3). Ceną



Rysunek 4.1: Udział powierzchni przechodniej w funkcji liczby kroków dla różnej liczby wędrowców.

za większą dostępność jest bardziej zawiła i relatywnie dłuższa trasa optymalna (rys. 4.4) co znacząco przekłada się na stopień trudności a tym samym na procent udanych wyjść z poziomu przez symulowanego gracza (rys. 4.2).

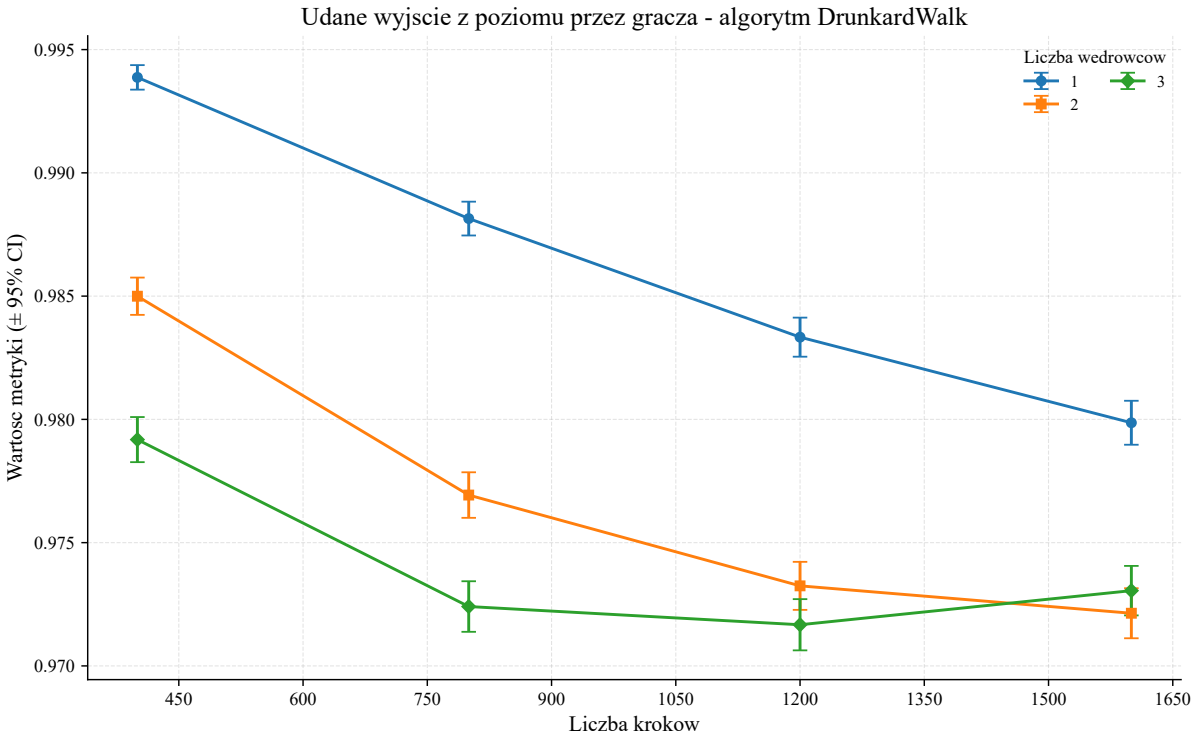
Preferencja kierunkowa a liniowość korytarzy.

Gdy wędrowiec częściej utrzymuje dotychczasowy kierunek, sieć przybiera bardziej tunelowy charakter. W całym badanym zakresie parametrów zależność jest prawie liniowa: każdy wzrost preferencji utrzymania dotychczasowego kierunku o $\Delta = 0{,}1$ dodaje rzędu $\approx 0{,}02{-}0{,}03$ punktu udziału stosunku korytarzy liniowych, najsilniej przy jednym wędrowcu (rys. 4.5).

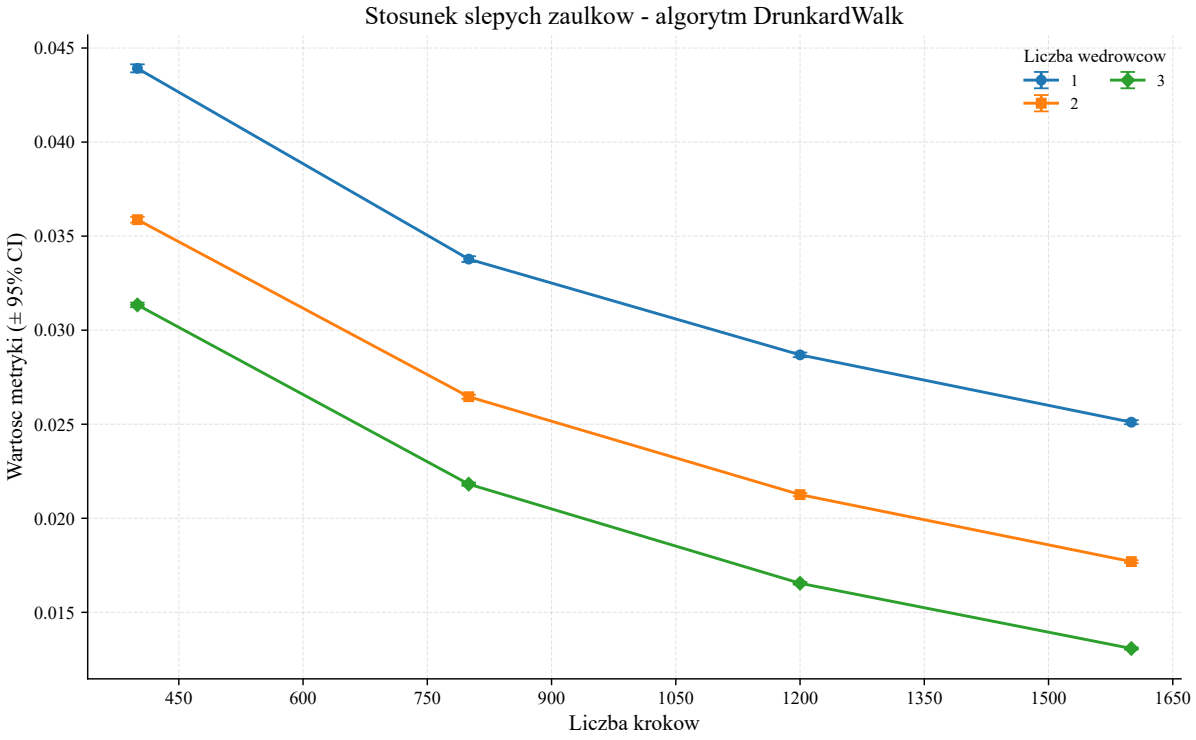
Dodatkowi wędrowcy ten efekt osłabiają (dla tych samych poziomów preferencji udział odcinków prostych jest niższy), a dłuższe przebiegi, nawet przy tej samej preferencji, prowadzą do bardziej poprzecinanej i krętej siatki. Intuicyjnie: im dłużej rysujemy i im więcej źródeł eksploruje jednocześnie, tym częściej przecinamy wcześniej wytyczone proste.

Wniosek cząstkowy.

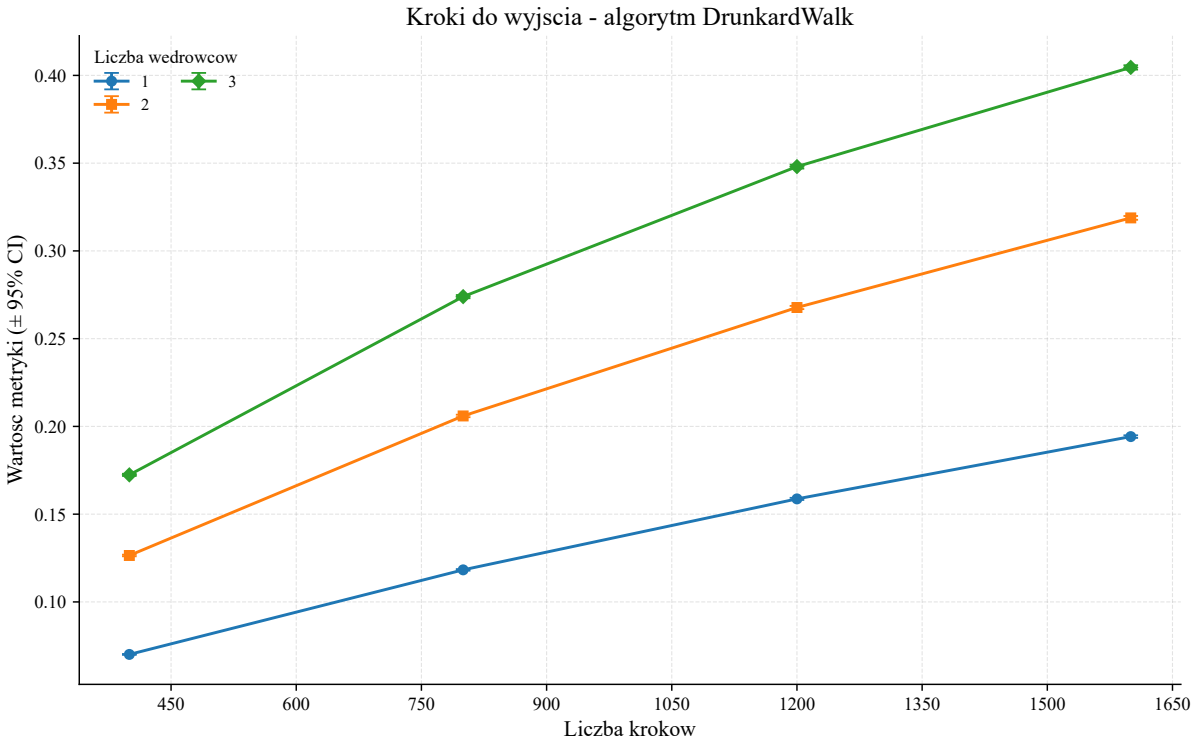
Zwiększanie liczby kroków systematycznie buduje przechodniość i spójność przestrzeni, a dokładanie kolejnych wędrowców przyspiesza te procesy. Jednocześnie wysoka preferencja kierunkowa zwiększa tendencje do powstawania tuneli, lecz wielośródłowa i długo-



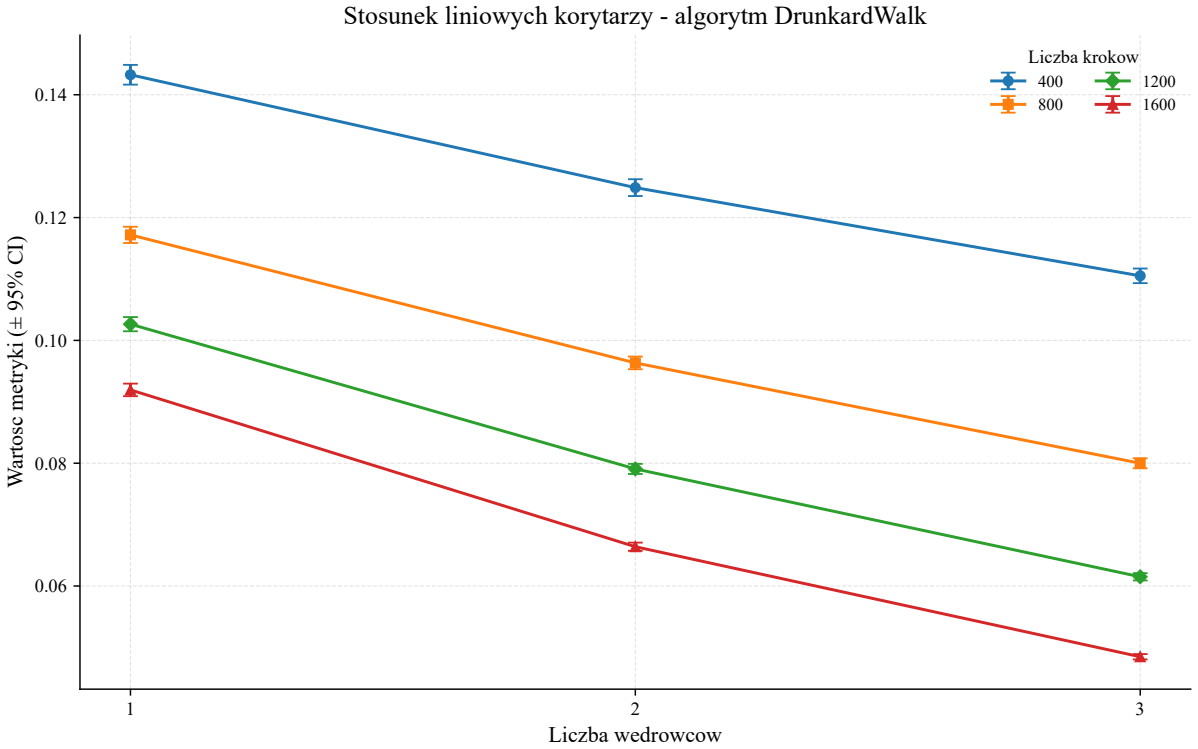
Rysunek 4.2: Procent udanych wyjść z poziomu przez gracza w funkcji liczby kroków dla różnej liczby wędrowców.



Rysunek 4.3: Stosunek ślepych zaułków w funkcji liczby kroków dla różnej liczby wędrowców.



Rysunek 4.4: Relatywne kroki do wyjścia w funkcji liczby kroków dla różnej liczby wędrowców.



Rysunek 4.5: Stosunek liniowych korytarzy w funkcji liczby wędrowców dla różnej liczby kroków.

trwała eksploracja rozprasza tę prostotę.

Analiza wpływu parametrów na metryki.

Stopień korelacji pomiędzy parametrami a metrykami przedstawia rys. 4.6. Najważniejsze wnioski jakie można wyciągnąć z trendów współczynnika Spearmana ρ dla tych które są istotne na poziomie $p < 0{,}05$, są następujące:

- **Liczba kroków.** Większa liczba kroków wiąże się dodatnio z przechodnią powierzchnią ($\rho \approx 0{,}68$) i średnim stopniem węzła ($\rho \approx 0{,}63$) oraz ujemnie ze ślepymi zaułkami ($\rho \approx -0{,}78$) i stagnacjami gracza ($\rho \approx -0{,}80$). Jednocześnie rośnie wskaźnik zakrętów ($\rho \approx 0{,}36$) i liczba kroków do wyjścia ($\rho \approx 0{,}66$).

- **Liczba wędrowców.** Większa liczba wędrowców wzmacnia łączność (dodatnia zależność z przechodnią powierzchnią, $\rho \approx 0{,}82$) i „prostuje" eksplorację w sensie równomiernego wypełniania mapy (silnie ujemna zależność z przesunięciem centroidu poziomu, $\rho \approx -0{,}90$). Jednocześnie maleje udział ślepych zaułków ($\rho \approx -0{,}57$), a rośnie tempo odkrywania ($\rho \approx 0{,}66$) oraz relatywna długość drogi do wyjścia ($\rho \approx 0{,}66$).

- **Preferencja kierunkowa.** Silna preferencja kierunku wyraźnie zwiększa udział odcinków prostych ($\rho \approx 0{,}87$) i — paradoksalnie — liczbę zmian kierunku na optymalnej ścieżce ($\rho \approx 0{,}75$; efekt „schodkowania" trasy w pobliżu przeszkód). Jednocześnie obniża średni stopień węzła ($\rho \approx -0{,}51$) oraz odsetek udanych wyjść i zdobytych kluczy (oba $\rho \approx -0{,}57$), a także ogólny procent odkrytego poziomu ($\rho \approx -0{,}37$). ($p < 0{,}001$).

- **Prawdopodobieństwo kroku wstecz.** Efekty są niewielkie: obserwujemy drobne osłabienie liniowości korytarzy ($\rho \approx -0{,}23$). Pozostałe zależności są słabe lub nieistotne.

Wnioski cząstkowe.

1. **Więcej kroków oznacza pełniejszą i czytelniejsza sieć przejść.** Zwiększanie liczby kroków systematycznie podnosi udział terenu możliwego do przejścia i średni stopień węzła, a ogranicza ślepe zaułki. Trend byłby utrzymany aż do pełnego zapełnienia powierzchni mapy terenem możliwym do przejścia. Jednocześnie rośnie liczba skrętów na trasie i długość ścieżki do wyjścia, co należy uwzględnić przy projektowaniu wyzwań dla gracza.

2. **Więcej wędrowców przyspiesza „wyrzeźbienie" planu, ale wydłuża dojście do celu.** Dodatkowi wędrowcy zwiększają przechodnią powierzchnię i ograniczają

lokalne pułapki (ślepe zaułki), jednak relatywna długość optymalnej ścieżki rośnie, a średnie odkrycie na krok spada — to naturalny koszt większego rozgałęzienia korytarzy. W ramach badanych wartości liczba korytarzy liniowych spada wraz z większą ilością wędrowców i kroków.

3. **Silna preferencja kierunkowa nie sprzyja grywalności.** Choć zwiększa udział odcinków prostych, jednocześnie pogarsza wyniki „zadaniowe" (mniej udanych wyjść, mniej zdobytych kluczy) i bywa źródłem „schodkowania" trasy. Zalecana jest wartość niska lub zerowa (Zależności z rys. 4.6).

4. **Krok wstecz ma znaczenie drugorzędne.** W rozważanym zakresie wartości jego wpływ jest niewielki; można go traktować jako parametr strojenia „smaku" korytarzy (odrobinę mniej odcinków prostych) bez dużego wpływu na kluczowe metryki.

4.3 Algorytm *Binarnego Podziału Przestrzeni*, BSP – studium przypadku

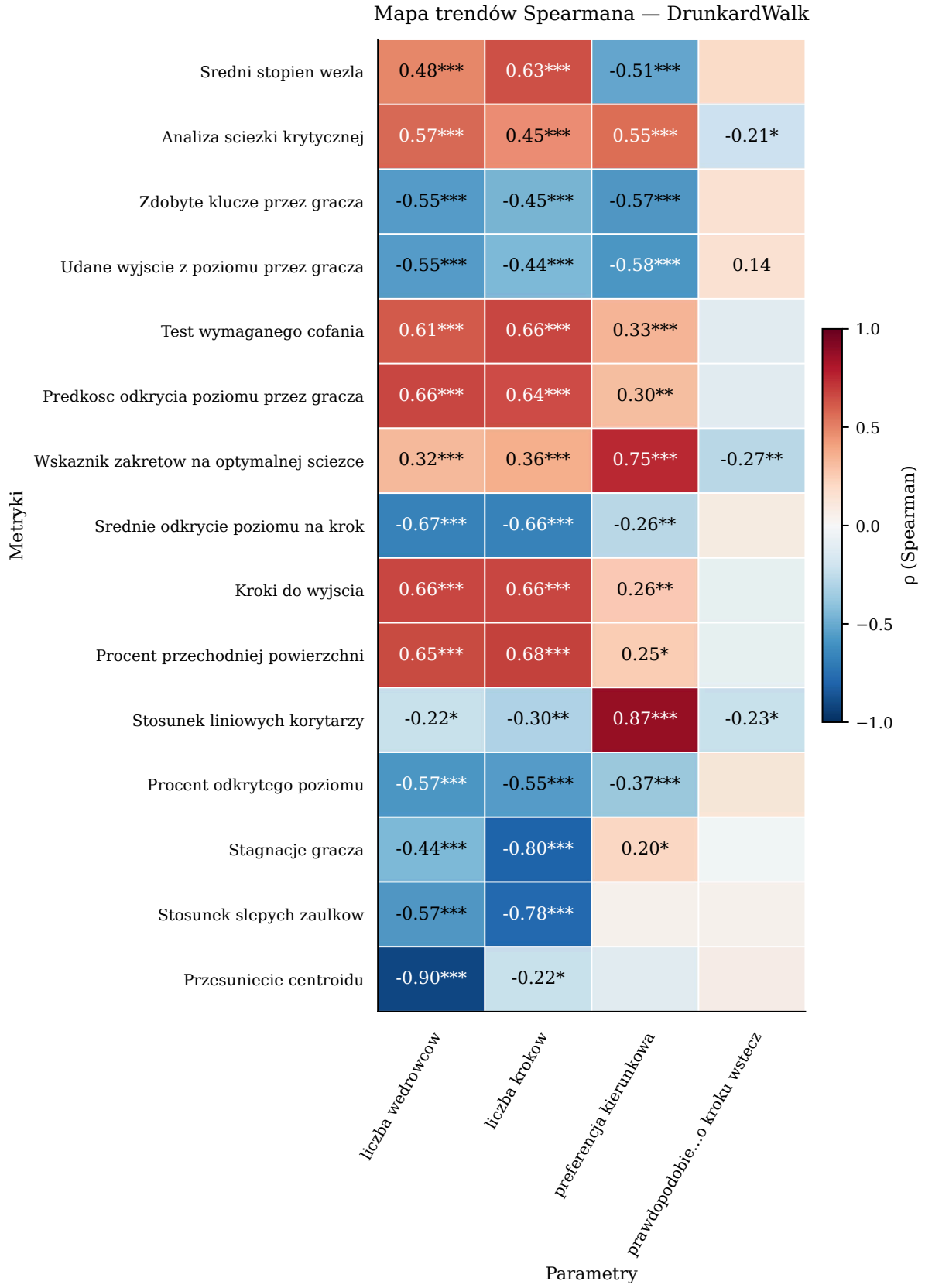
Analiza uwzględnia przegląd głównych parametrów algorytmu *Binarnego Podziału Przestrzeni*: *maksymalna głębokość podziału*, *minimalny rozmiar komnaty*, *minimalny wymiar regionu do podziału*, *szerokość korytarza* oraz *liczbę dodatkowych połączeń* pomiędzy pomieszczeniami. Wyniki poddano analizie korelacji Spearmana, a istotne wartości uwzględniono przy poziomie $p < 0{,}05$. Łącznie przeanalizowano 108 konfiguracji, co odpowiada około $6{,}4 \times 10^6$ pojedynczym przebiegom symulacji gracza. Plik *data_BSP.json* z wynikami eksperymentów dołączono do pracy w załączniku.

Powierzchnia przechodnia a głębokość podziału i rozmiar komnaty.

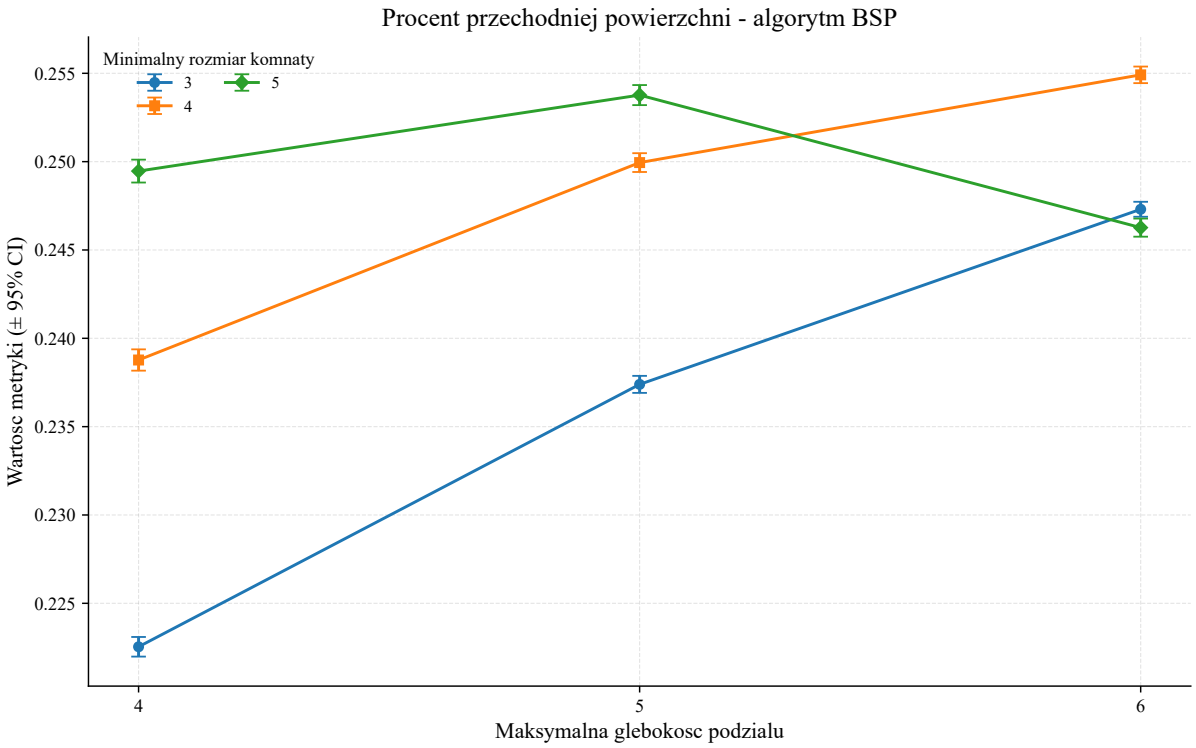
Wraz ze wzrostem *maksymalnej głębokości podziału* rośnie udział powierzchni możliwej do przejścia. W dolnym zakresie konfiguracji obserwujemy wartości rzędu $\sim 0{,}215$, podczas gdy w konfiguracjach z głębszym podziałem lub większymi komnatami udział ten dochodzi do $\sim 0{,}254$. *Minimalny rozmiar komnaty* dodatkowo wzmacnia tę tendencję: większe pokoje sprzyjają lepszej łączności i rozpraszają wąskie gardła typowe dla bardzo drobnego podziału (rys. 4.7). Przy ekstremalnych ustawieniach (duże komnaty i maksymalna głębokość) wzrost się odwraca, co sugeruje, że niektóre pokoje nie mają warunków lub miejsca na powstanie, ograniczając tym samym powierzchnię poziomu.

Długość trasy i struktura korytarzy.

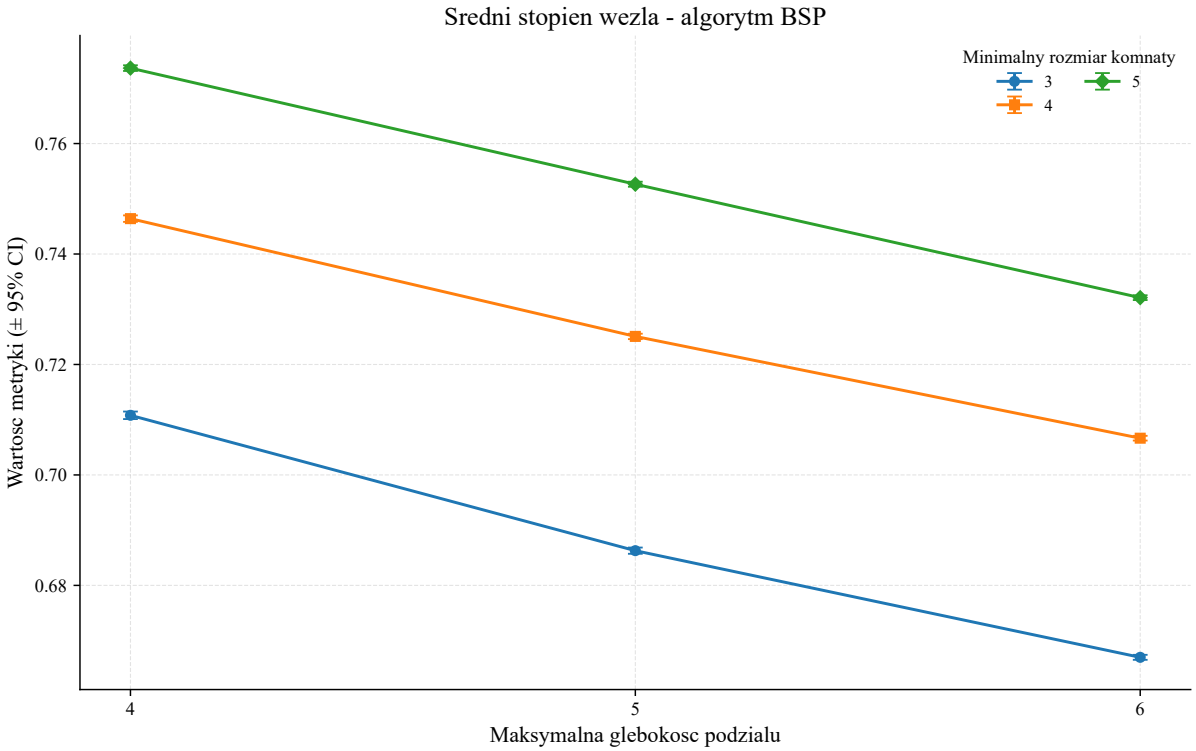
Większe komnaty istotnie zwiększają *średni stopień węzła* (od $\sim 0{,}658$ do $\sim 0{,}781$) i redukują *stosunek liniowych korytarzy* (z $\sim 0{,}225$ do $\sim 0{,}121$), co oznacza bogatszą sieć



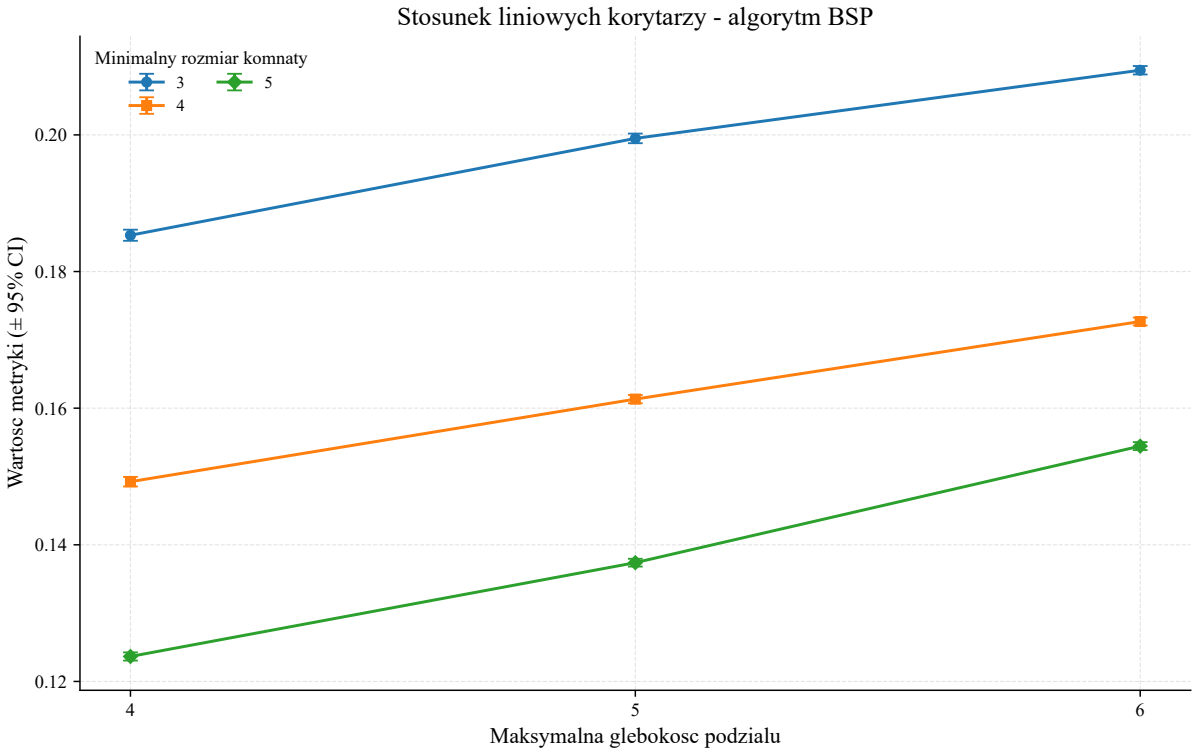
Rysunek 4.6: Korelacje rangowe Spearmana między metrykami a parametrami algorytmu *Pijany Spacer* (komórki: ρ). Istotność: * $p < 0{,}05$; ** $p < 0{,}01$; *** $p < 0{,}001$. Adnotacje ograniczono do relacji istotnych ($p < 0{,}05$) lub o sile $|\rho| \geq 0{,}10$.



Rysunek 4.7: Udział powierzchni przechodniej w funkcji maksymalnej głębokości podziału dla różnego minimalnego rozmiaru komnaty.



Rysunek 4.8: Średni stopień węzła w funkcji maksymalnej głębokości podziału dla różnego minimalnego rozmiaru komnaty.



Rysunek 4.9: Stosunek liniowych korytarzy w funkcji maksymalnej głębokości podziału dla różnego minimalnego rozmiaru komnaty.

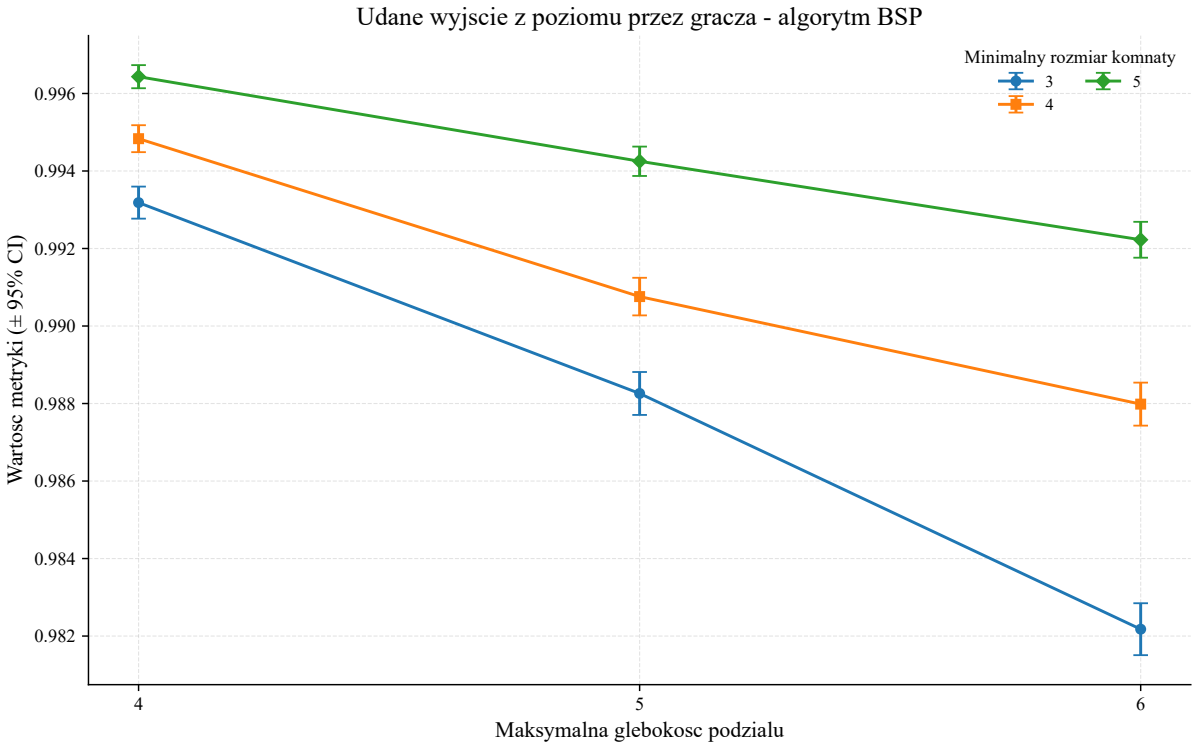
połączeń i mniejszą „tunelowość" i większą przestrzeń (rys. 4.8 i 4.9). W badanym zaobserwowano również fakt, że *stosunek ślepych zaułków* wynosi praktycznie 0 dla wszystkich konfiguracji, oznacza to, że algorytm nie tworzy „martwych końców".

Skuteczność i „płynność" poruszania.

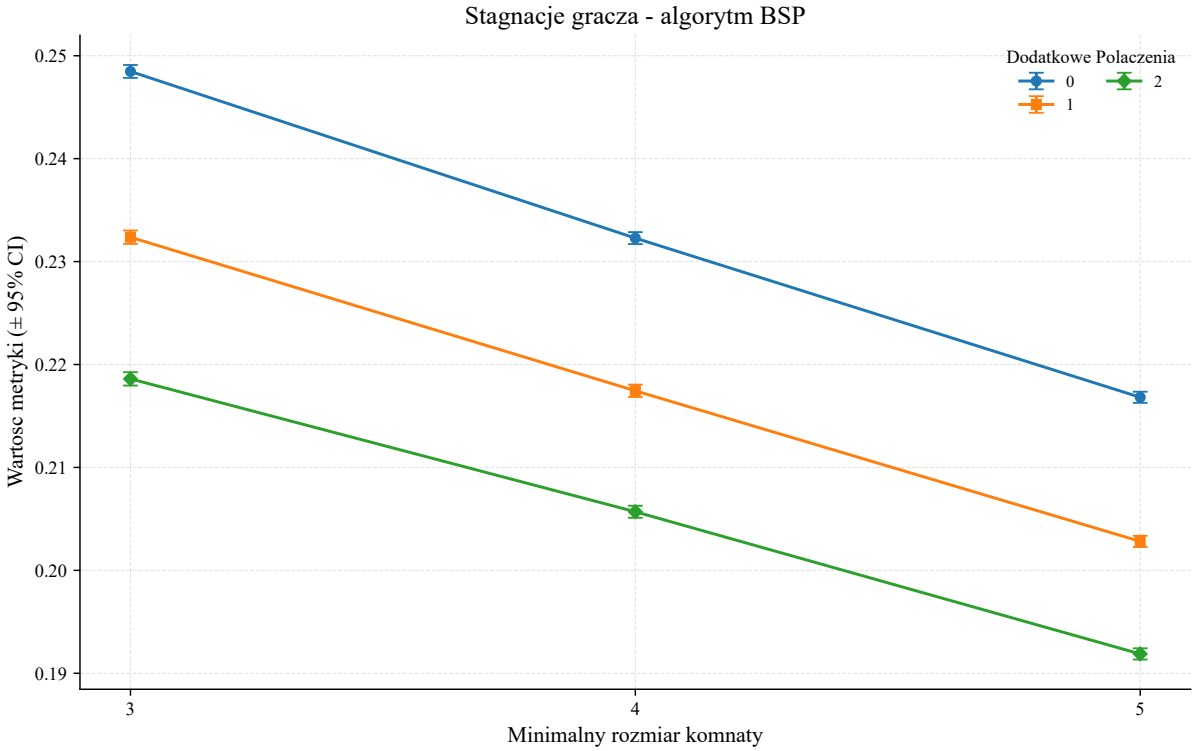
W całym przekroju metryki zadaniowe są bardzo wysokie: *procent udanych wyjść* oscyluje wokół $\sim 0{,}99$, a *procent zdobytych kluczy* jest bliski jedności. Głębszy podział ma tendencję do obniżania tych wskaźników, natomiast większe komnaty, do ich poprawy (rys. 4.10). *Procent odkrytego poziomu* utrzymuje się natomiast blisko $\sim 0{,}671$. Dodatkowo wraz z pogłębianiem podziału rosną *stagnacje gracza* (zakres $\sim 0{,}176{-}0{,}273$) a dodatkowe połączenia i większe komnaty przeciwdziałają stagnacjom co pokazuje rys. 4.11.

Analiza wpływu parametrów na metryki.

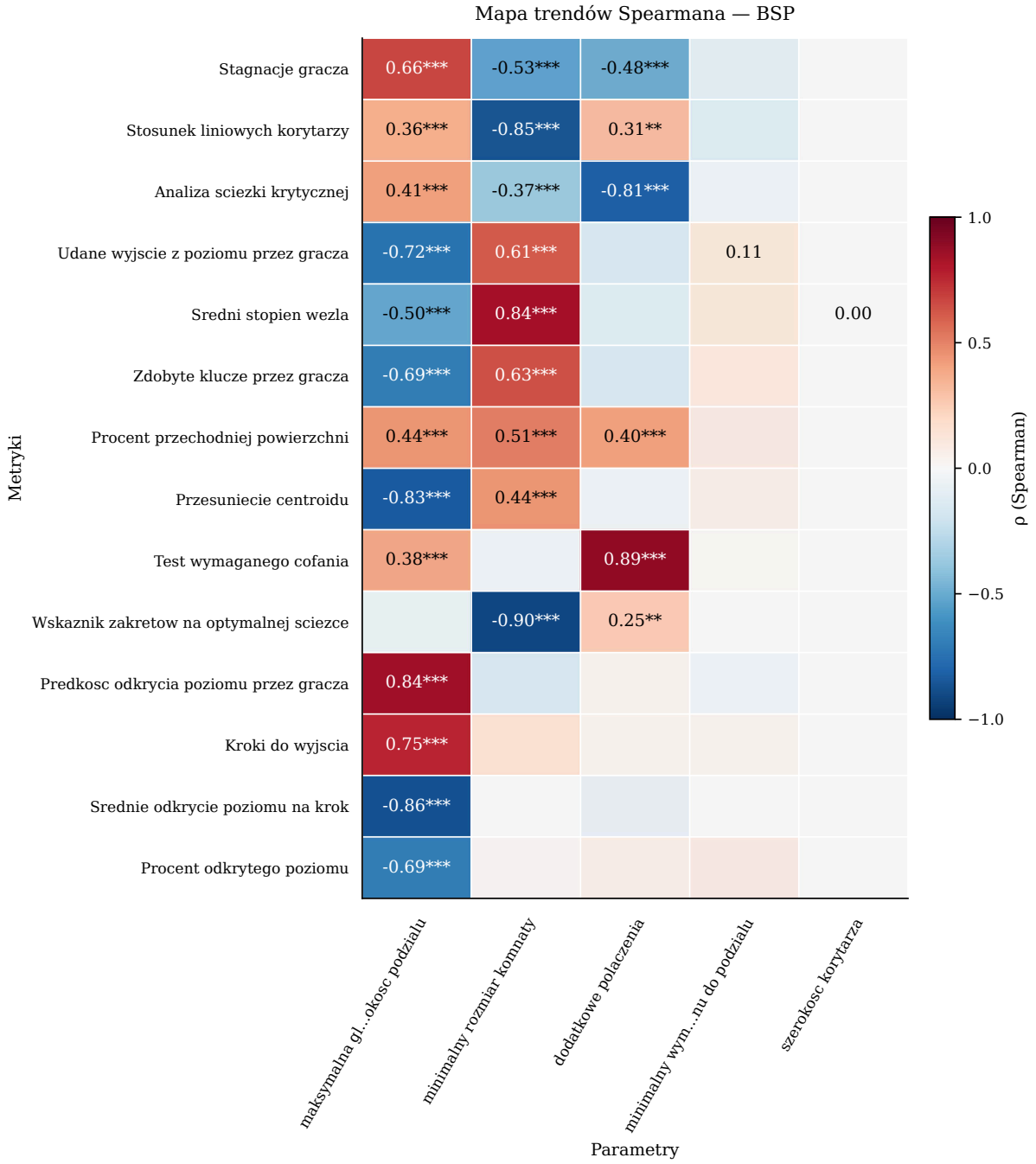
Rys. 4.12 prezentuje współczynniki Spearmana ρ (istotne dla $p < 0{,}05$) i pozwala zarysować ogólny obraz zależności między parametrami a metrykami. Wraz z pogłębianiem podziału przestrzeni rośnie udział powierzchni możliwej do przejścia ($\rho \approx 0{,}442$), ale jednocześnie wydłuża się optymalna trasa do wyjścia ($\rho \approx 0{,}752$). Ten sam mechanizm „zagęszczania" struktury ma też koszt: maleje procent odkrytego poziomu ($\rho \approx -0{,}693$) i



Rysunek 4.10: Procent udanych wyjść z poziomu przez gracza w funkcji maksymalnej głębokości podziału dla różnego minimalnego rozmiaru komnaty.



Rysunek 4.11: Stagnacje gracza w funkcji minimalnego rozmiaru dla różnej ilości dodatkowych połączeń.



Rysunek 4.12: Korelacje rangowe Spearmana między metrykami a parametrami algorytmu *BSP* (komórki: ρ). Istotność: * $p < 0{,}05$; ** $p < 0{,}01$; *** $p < 0{,}001$. Adnotacje ograniczono do relacji istotnych ($p < 0{,}05$) lub o sile $|\rho| \geq 0{,}10$.

spada odsetek udanych wyjść ($\rho \approx -0{,}721$), a dodatkowo częściej obserwujemy stagnacje gracza ($\rho \approx 0{,}664$). Z kolei zwiększanie minimalnego rozmiaru komnaty działa porządkująco na nawigację: wzrasta średni stopień węzła ($\rho \approx 0{,}835$), maleje tunelowość korytarzy ($\rho \approx -0{,}854$) i liczba zakrętów na trasie ($\rho \approx -0{,}899$), co przekłada się na więcej udanych wyjść z poziomu ($\rho \approx 0{,}608$) i liczbę zdobytych kluczy ($\rho \approx 0{,}632$). W przeciwieństwie do tych dwóch osi strojenia, minimalny wymiar regionu do podziału okazuje się w badanym zakresie parametrem drugorzędnym: jego wpływ na metryki jest niewielki lub statystycznie nieistotny. Uzupełniając, dokładanie dodatkowych połączeń między komnatami skraca ścieżkę krytyczną ($\rho \approx -0{,}810$) i zmniejsza liczbę stagnacji ($\rho \approx -0{,}485$), choć jednocześnie umiarkowanie podbija liniowość korytarzy ($\rho \approx 0{,}308$). Innymi słowy: głębokość podziału „dodaje treści" kosztem kosztu przejścia, wielkość komnat porządkuje przepływ ruchu i ułatwia zadanie graczowi, a dodatkowe korytarze są skutecznym sposobem na polepszenie grywalności bez naruszania ogólnej struktury.

Wnioski cząstkowe.

1. **Głębszy podział to więcej treści, ale dłuższe dojście do celu.** Rośnie przechodniość i kroki do wyjścia, maleje procent odkrycia i lekko spada odsetek udanych wyjść, a stagnacje rosną.

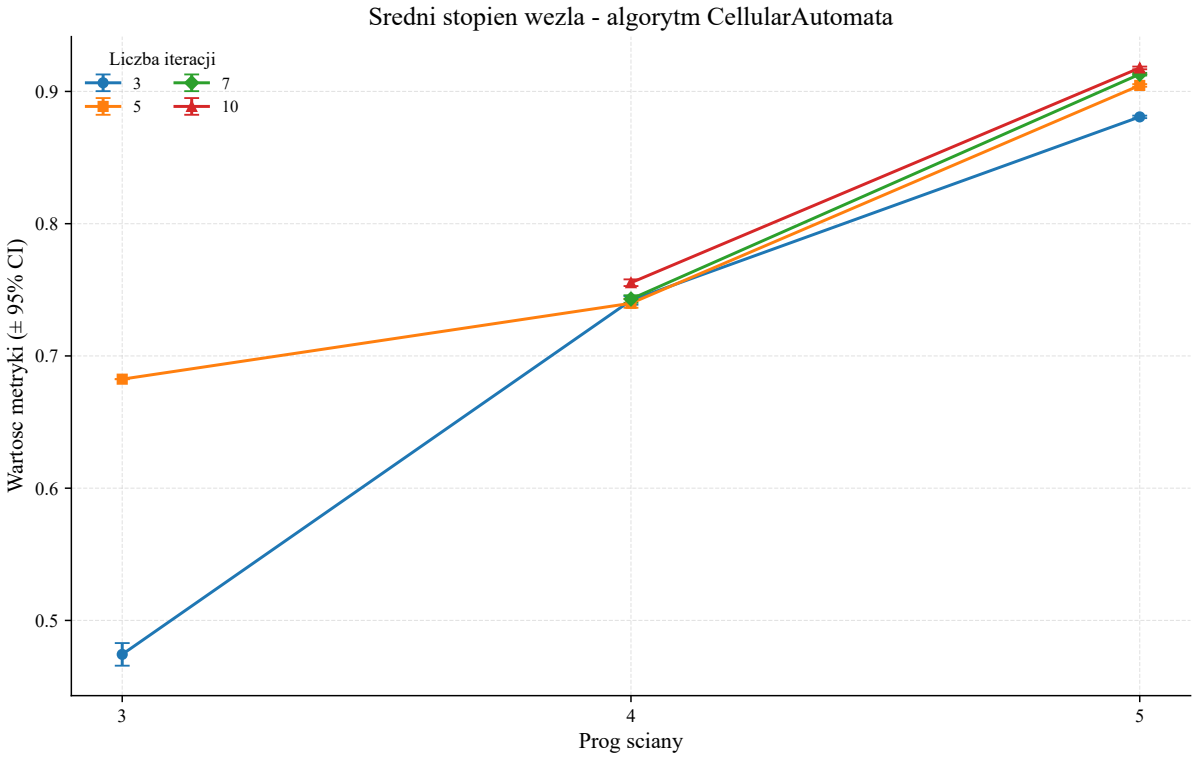
2. **Większe komnaty czynią nawigację bardziej czytelną.** Zauważamy dużo większą otwartość terenu, obniża się również tunelowość i liczbę zakrętów, podnosząc skuteczność przejścia.

3. **Brak ślepych zaułków.** W testowanym zakresie parametrów stosunek ślepych zaułków ≈ 0, co świadczy o braku lokalnych pułapek topologicznych.

4. **Kilka dodatkowych korytarzy może wpływać pozytywnie na grywalność.** Dodanie 1–2 dodatkowych korytarzy między komnatami skraca ścieżkę krytyczną i obniża stagnacje bez istotnego kosztu w innych metrykach (poza wzrostem w wymaganym cofaniu przez gracza, które w zależności od kontekstu, może być traktowane pozytywnie lub negatywnie).

4.4 Algorytm *Automatu Komórkowego*, CellularAutomata – studium przypadku

Analizie poddano cztery główne algorytmu *Automatu Komórkowego*: *prawdopodobieństwo ściany*, *liczbę iteracji wygładzania*, *próg ściany* oraz *próg podłogi*. Wyniki poddano analizie korelacji Spearmana, a istotne wartości uwzględniono przy poziomie $p < 0{,}05$. Łącznie przeanalizowano 144 konfiguracji, co odpowiada około $8{,}64 \times 10^6$ pojedynczym



Rysunek 4.13: Średni stopień węzła w funkcji progu ściany dla różnej liczby iteracji wygładzania.

przebiegom symulacji gracza. Plik *data_CellularAutomata.json* z wynikami eksperymentów dołączono do pracy w załączniku.

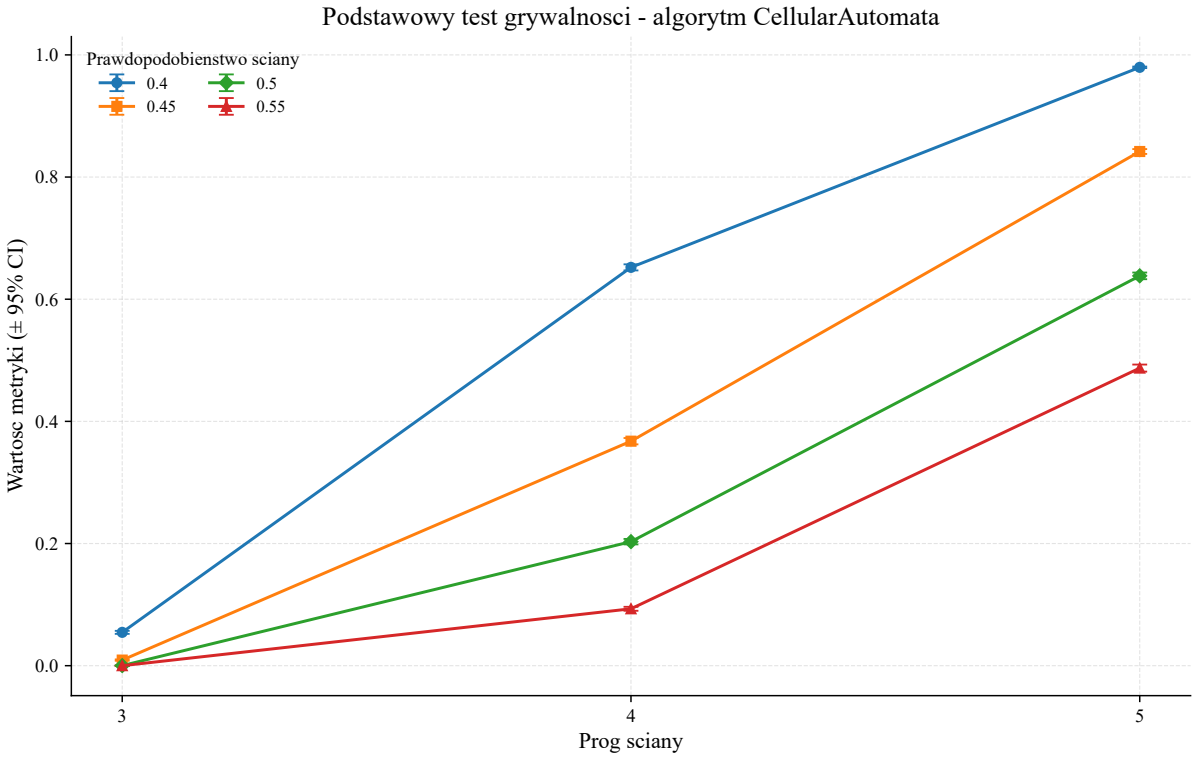
Średni stopień węzła a próg ściany i liczba iteracji wygładzania

Na rys. 4.13 pokazano zależność średniego stopnia węzła od *progu ściany* przy różnych wartościach *liczby iteracji*. Wraz ze wzrostem progu ściany wskaźnik systematycznie rośnie. Przy niskim progu (3) wartości pozostają niskie i zależą silnie od liczby iteracji (od około $\sim 0{,}47$ dla 3 iteracji do $\sim 0{,}68$ dla 5 iteracji). Przy wyższych progach (4−5) różnice pomiędzy iteracjami maleją, a średni stopień węzła osiąga wartości bliskie $\sim 0{,}91$.

Można więc zauważyć, że próg ściany jest głównym czynnikiem zwiększającym spójność układu, a dodatkowe iteracje wygładzania działają wspierająco, szczególnie przy niskich progach, gdzie ograniczają fragmentację i poprawiają łączność poziomu. Przy bardzo wysokim progu ściany i niekorzystnych ustawieniach pozostałych parametrów średni stopień węzła przestaje rosnąć.

Podstawowy test grywalności a parametry wejściowe

Podstawowy test grywalności, który mierzy odsetek udanych generacji grywalnego poziomu, w największym stopniu zależy od *progu ściany*. Przy niskiej wartości tego parametru skuteczność spada niemal do zera, ponieważ układ staje się zbyt rozdrobniony i trudny



Rysunek 4.14: Podstawowy test grywalności w funkcji progu ściany dla różnego prawdopodobieństwa ściany.

do przejścia. Przy progu ściany równym (4) można zaobserwować skuteczność od około $\sim 0{,}65$ dla małego prawdopodobieństwa ściany do około $\sim 0{,}12$ dla większej zabudowy (rys. 4.14). Najlepsze wyniki osiągane są przy wysokim progu (5), gdzie na początku skuteczność wynosi prawie $\sim 0{,}98$, a nawet przy wyższym zagęszczeniu spada tylko do około $\sim 0{,}48$. *Prawdopodobieństwo ściany* wpływa tutaj negatywnie. Im więcej ścian pojawia się w początkowym układzie, tym szybciej maleje odsetek udanych wyjść. Nawet gdy próg ściany jest korzystny, zbyt gęsta zabudowa ogranicza grywalność (rys. 4.14).

Wyniki wskazują, że dla tej metryki najważniejsze jest utrzymanie wysokiego progu ściany oraz umiarkowanego prawdopodobieństwa ściany. W optymalnych ustawieniach (próg 5 i prawdopodobieństwo w zakresie $0{,}4-0{,}45$) skuteczność przejścia zbliża się do 100%, co potwierdza stabilność algorytmu w tych warunkach.

Analiza wpływu parametrów na metryki

Analiza oparta na mapie korelacji Spearmana ($p < 0{,}05$) przedstawionej na rys. 4.15 pozwala uchwycić zależności pomiędzy parametrami wejściowymi a najważniejszymi metrykami. Najsilniej na kształt mapy działa *próg ściany*. Wraz ze wzrostem jego wartości rośnie udział powierzchni przechodniej ($\rho \approx 0{,}77$), a także wydłuża się optymalna ścieżka do wyjścia ($\rho \approx 0{,}72$). Wyższy próg zmniejsza też liczbę ślepych zaułków ($\rho \approx -0{,}83$) i ogranicza liniowość korytarzy ($\rho \approx -0{,}55$), co sprawia, że poziom staje się bardziej spójny

i zróżnicowany.

Liczba iteracji wygładzania ma mniejszy, ale zauważalny wpływ. Iteracje wydłużają trasę ($\rho \approx 0{,}21$), zwiększają liczbę zakrętów ($\rho \approx 0{,}34$) i poprawiają wskaźniki zadaniowe, takie jak odsetek udanych wyjść czy zdobytych kluczy ($\rho \approx 0{,}35$). Co ważne, iteracje zmniejszają stagnacje gracza ($\rho \approx -0{,}34$), co pozytywnie wpływa na płynność rozgrywki. *Próg podłogi* dodatkowo wspiera grywalność. Zwiększa skuteczność wyjść i zdobywania kluczy ($\rho \approx 0{,}33{-}0{,}34$), a także sprzyja eksploracji: procent odkrytej mapy ($\rho \approx 0{,}35$) i prędkość odkrywania ($\rho \approx 0{,}38$) rosną wraz z jego wartością. Odmiennie działa *prawdopodobieństwo ściany*. Wyższe wartości tego parametru zmniejszają powierzchnię przechodnią ($\rho \approx -0{,}28$), skracają ścieżkę do wyjścia ($\rho \approx -0{,}34$), ale równocześnie zwiększają stagnacje gracza ($\rho \approx 0{,}24$) i obniżają procent odkrycia mapy ($\rho \approx -0{,}27$).

Podsumowując, próg ściany nadaje globalny charakter poziomu, iteracje i próg podłogi poprawiają płynność i grywalność, a wysokie prawdopodobieństwo ściany komplikuje eksplorację i ogranicza odkrywanie mapy.

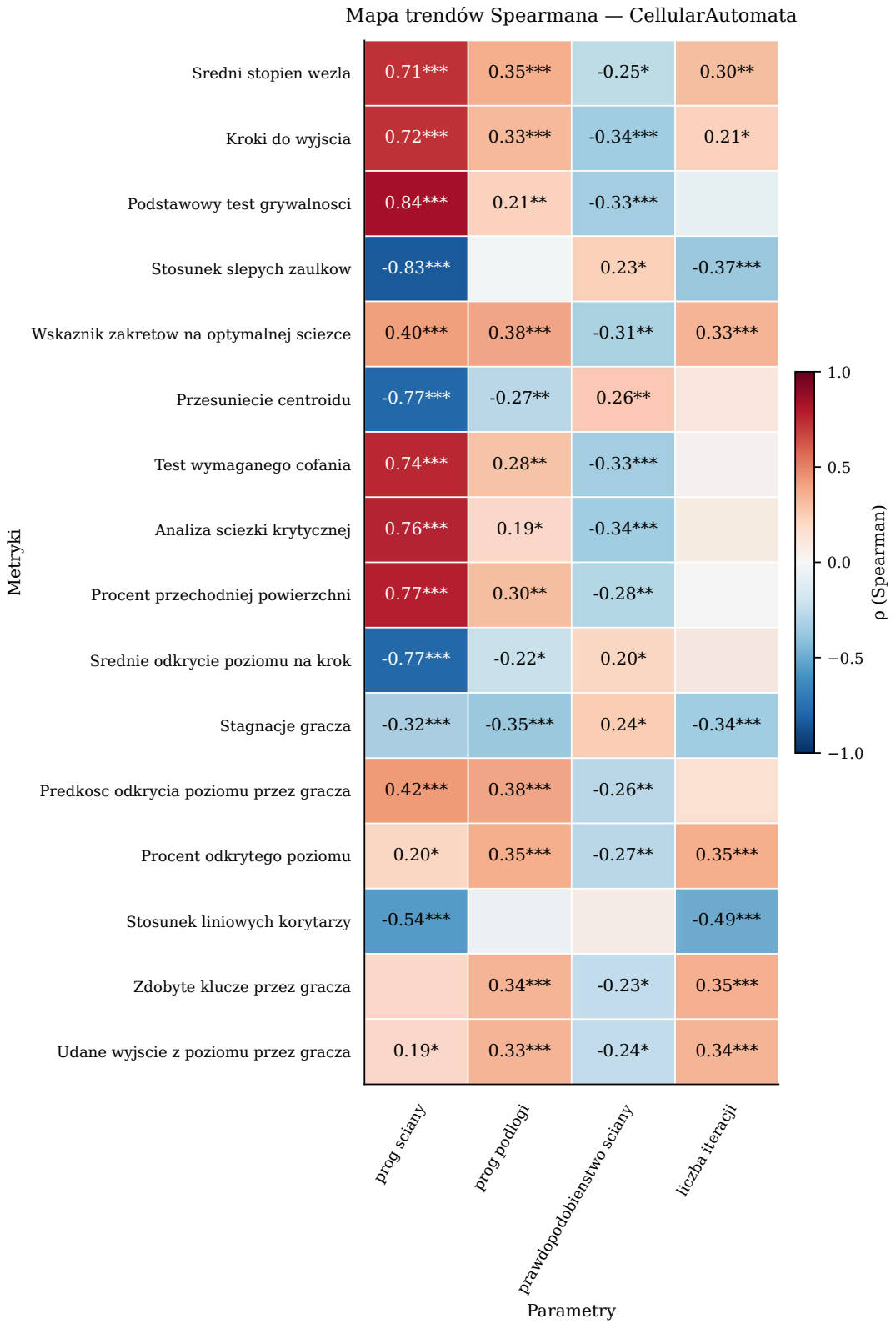
Wnioski cząstkowe. Analiza wskazuje, że algorytm *Automatów Komórkowych* pozwala tworzyć mapy o wysokiej spójności i organicznym charakterze, w których płynność gry zależy od wyważenia parametrów. Iteracje wygładzania poprawiają grywalność i ograniczają stagnacje, próg podłogi sprzyja dostępności i odkrywaniu, a próg ściany wzmacnia globalną spójność kosztem wydłużenia trasy i wzrostu liczby zakrętów. Prawdopodobieństwo ściany działa w odwrotnym kierunku, obniżając przechodniość i zwiększając stagnacje.

4.5 Algorytm *Agregacji Ograniczonej Dyfuzją*, DLA – studium przypadku

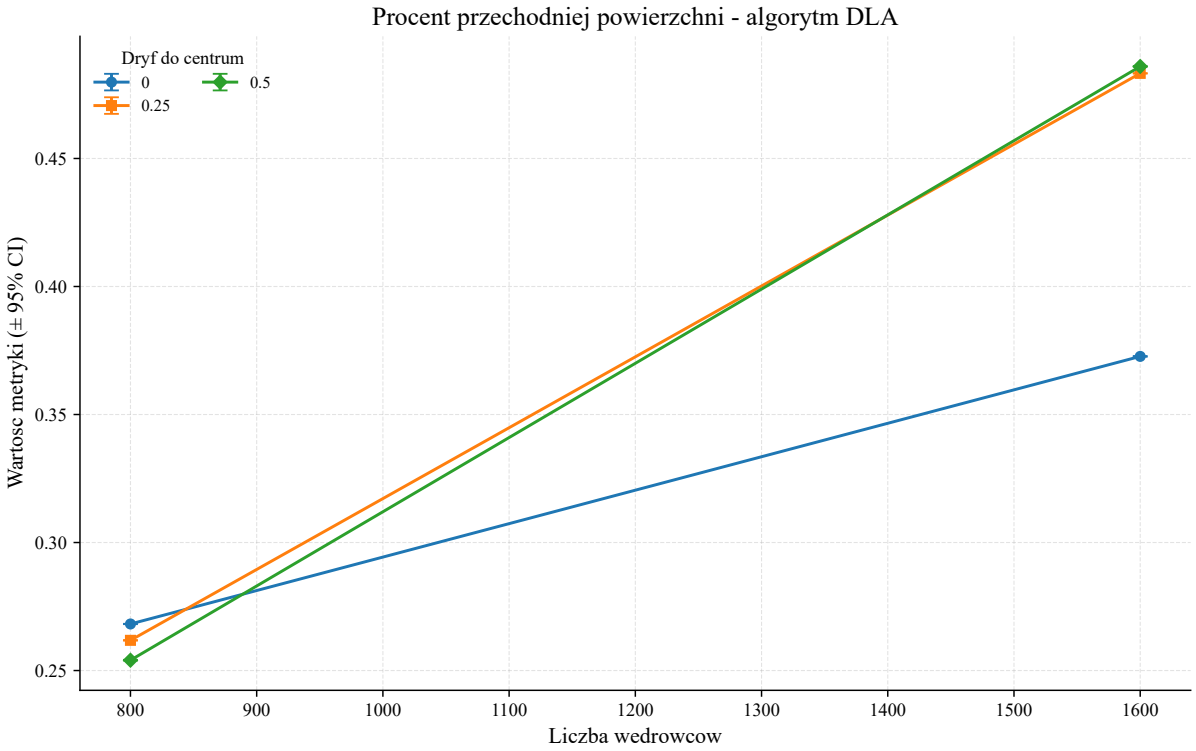
W tej części pracy przedstawiono wyniki analizy ilościowej wpływu parametrów algorytmu *Agregacji Ograniczonej Dyfuzją* na cechy topologiczne i grywalnościowe generowanych poziomów. Badania obejmowały pięć parametrów: *liczbę wędrowców*, *dryf do centrum*, *prawdopodobieństwo przylegania*, *margines startu* oraz *margines usunięcia*. Wyniki poddano analizie korelacji Spearmana, a istotne wartości uwzględniono przy poziomie $p < 0{,}05$. Łącznie przeanalizowano 108 konfiguracji, co odpowiada około $6{,}4 \times 10^6$ pojedynczym przebiegom symulacji gracza. Plik *data_DLA.json* z wynikami eksperymentów dołączono do pracy w załączniku.

Powierzchnia przechodnia a liczba wędrowców i dryf do centrum.

Większa *liczba wędrowców* systematycznie zwiększa przechodniość oraz łączność układu, co widać na rys. 4.16. Dla wysokiego dryfu do centrum 0,5 przejście z 800 do 1600 wędrowców podnosi udział powierzchni możliwej do przejścia z $\sim 0{,}26$ do $\sim 0{,}47$ (rys. 4.16).



Rysunek 4.15: Korelacje rangowe Spearmana między metrykami a parametrami algorytmu *Automatu Komórkowego* (komórki: ρ). Istotność: * $p < 0{,}05$; ** $p < 0{,}01$; *** $p < 0{,}001$. Adnotacje ograniczono do relacji istotnych ($p < 0{,}05$) lub o sile $|\rho| \geq 0{,}10$.



Rysunek 4.16: Udział powierzchni przechodniej w funkcji liczby wędrowców dla różnych poziomów dryfu do centrum

Jednocześnie wydłuża się trasa do wyjścia z $\sim 0{,}24$ do $\sim 0{,}33$ (rys. 4.18) oraz spada odsetek udanych wyjść z $\sim 0{,}88$ do $\sim 0{,}69$ (rys. 4.17).

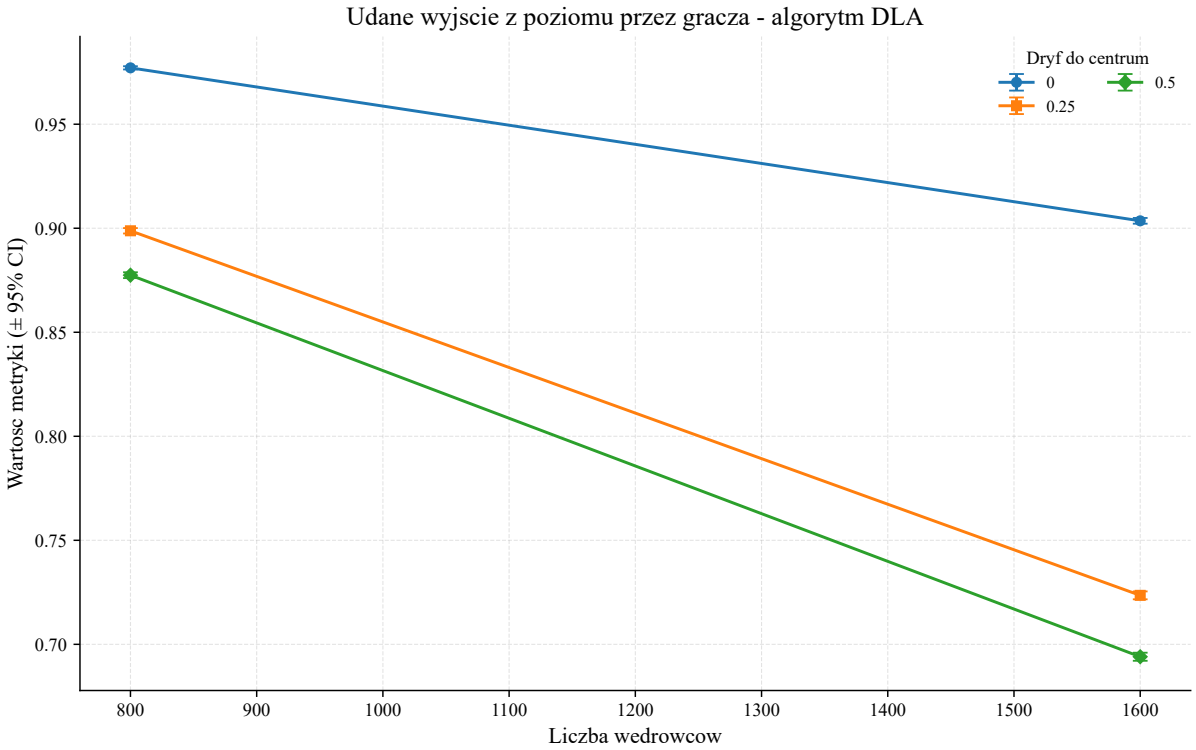
Wzrost *dryfu do centrum* zacieśnia eksplorację wokół środka planszy. *Kroki do wyjścia* są niższe dla większego dryfu przy tej samej liczbie wędrowców (zielona krzywa na rys. 4.18) natomiast maleje *procent udanych wyjść* (rys. 4.17) oraz *procent odkrytego poziomu* względem wariantów o słabszym dryfie.

Przyleganie a liniowość korytarzy.

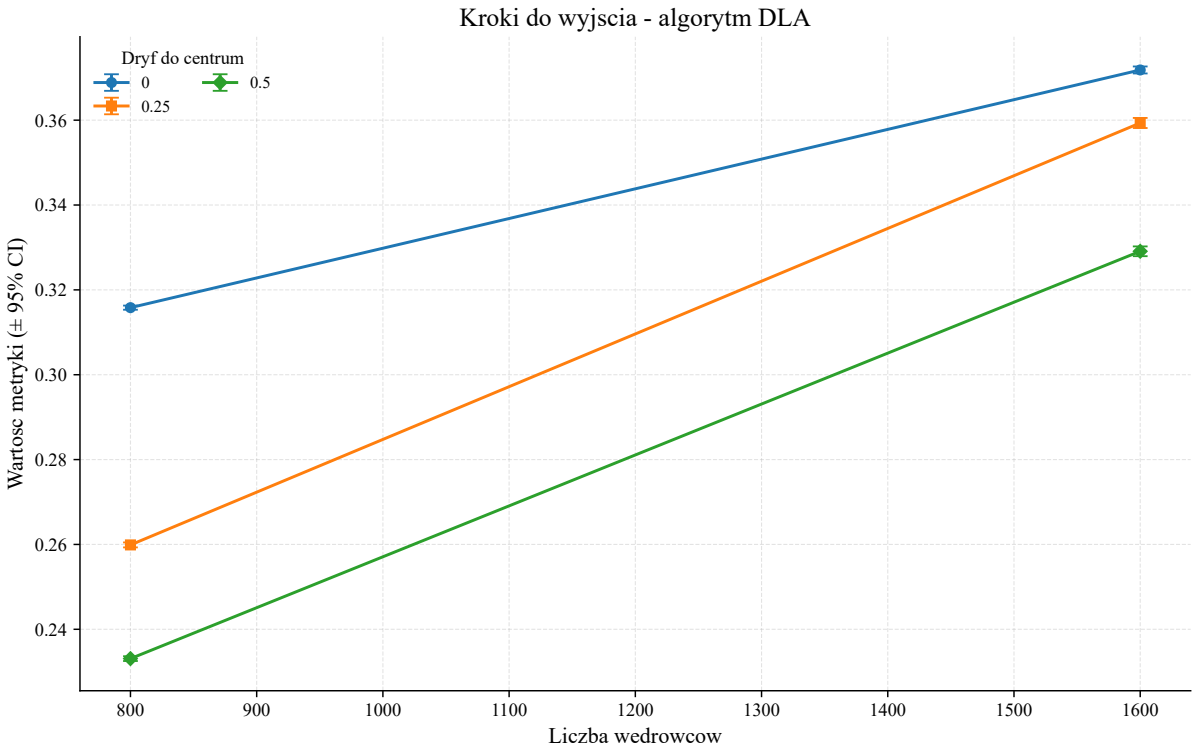
Wzrost *prawdopodobieństwa przylegania* podbija *stosunek korytarzy liniowych*, co można zaobserwować na rys. 4.19. Dla 800 wędrowców wartość rośnie od $\sim 0{,}21$ przy 0,8 do $\sim 0{,}23$ przy 1,0. Dla 1600 wędrowców wzrost jest silniejszy od $\sim 0{,}20$ do $\sim 0{,}26$ (rys. 4.19). Jednocześnie spada *średni stopień węzła*. Dla 800 wędrowców obniża się z $\sim 0{,}48$ do $\sim 0{,}40$ a dla 1600 wędrowców z $\sim 0{,}58$ do $\sim 0{,}51$ (rys. 4.20). Rośnie również częstość *stagnacji gracza*. Dla 800 wędrowców z $\sim 0{,}42$ do $\sim 0{,}44$ a dla 1600 wędrowców z $\sim 0{,}45$ do $\sim 0{,}47$ (rys. 4.21).

Analiza wpływu parametrów na metryki

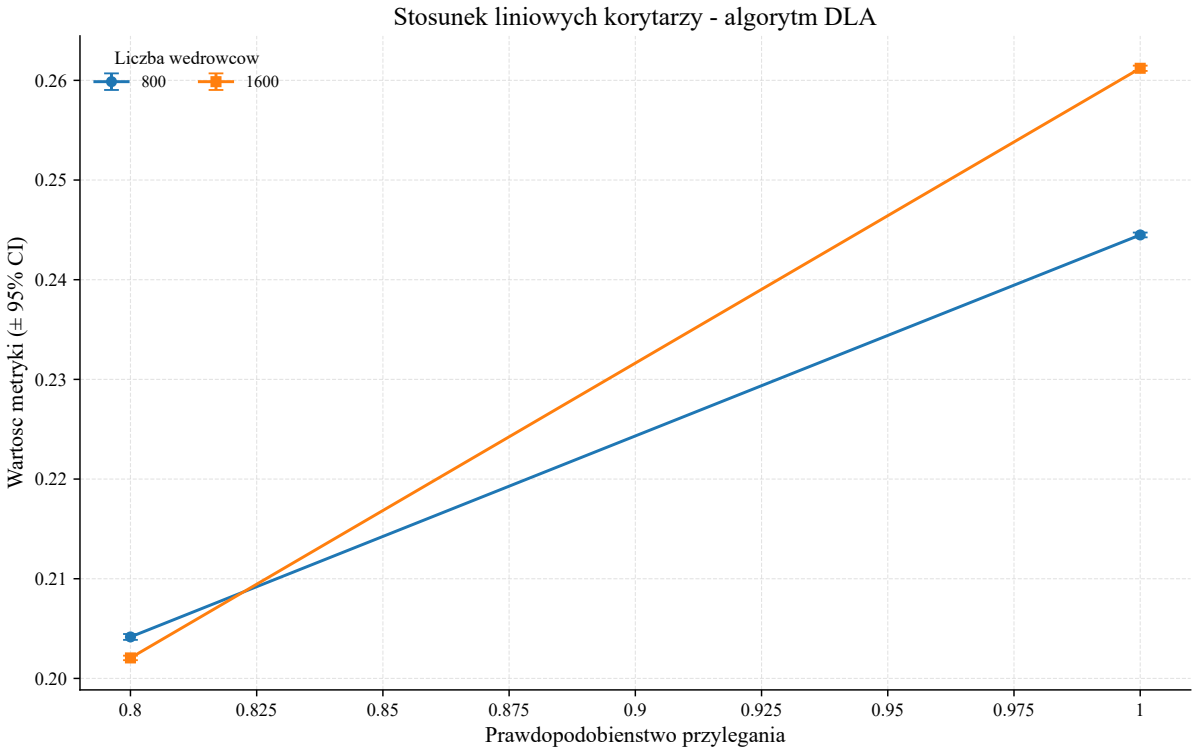
Rys. 4.22 prezentuje współczynniki Spearmana ρ (istotne dla $p < 0{,}05$) i pozwala zarysować ogólny obraz zależności między parametrami a metrykami. Najsilniejszy wpływ ob-



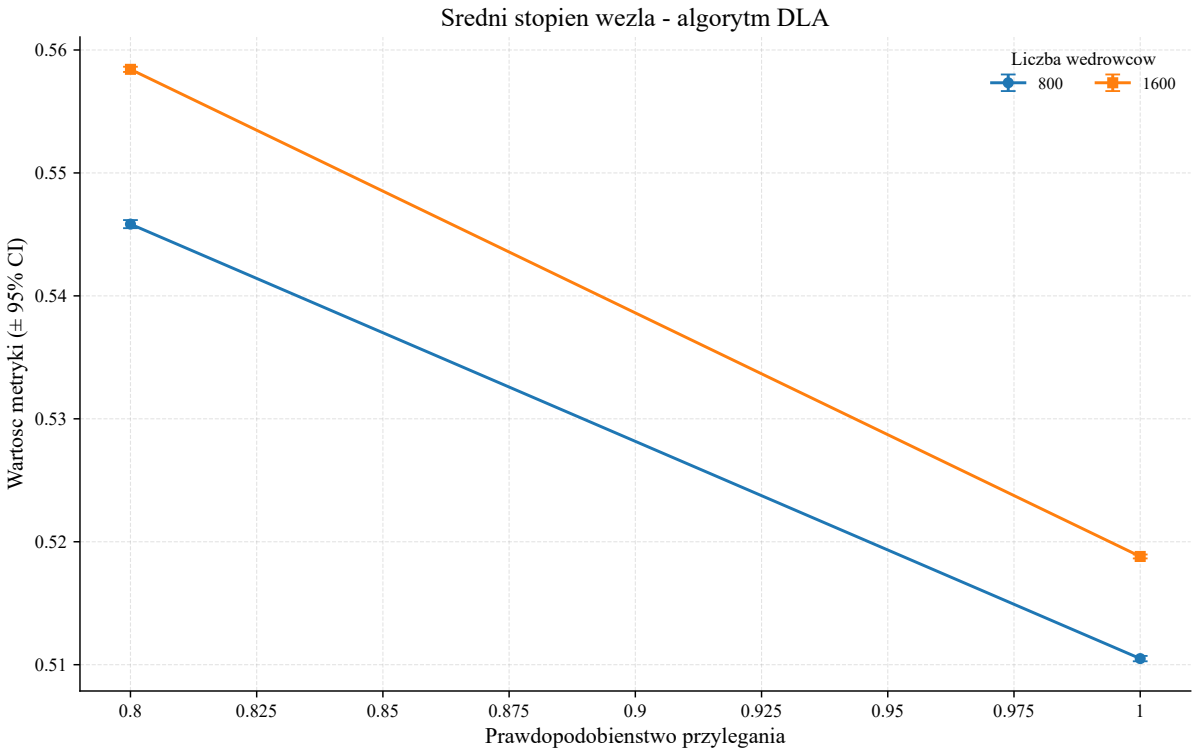
Rysunek 4.17: Procent udanych wyjść z poziomu w funkcji liczby wędrowców dla różnych poziomów dryfu do centrum



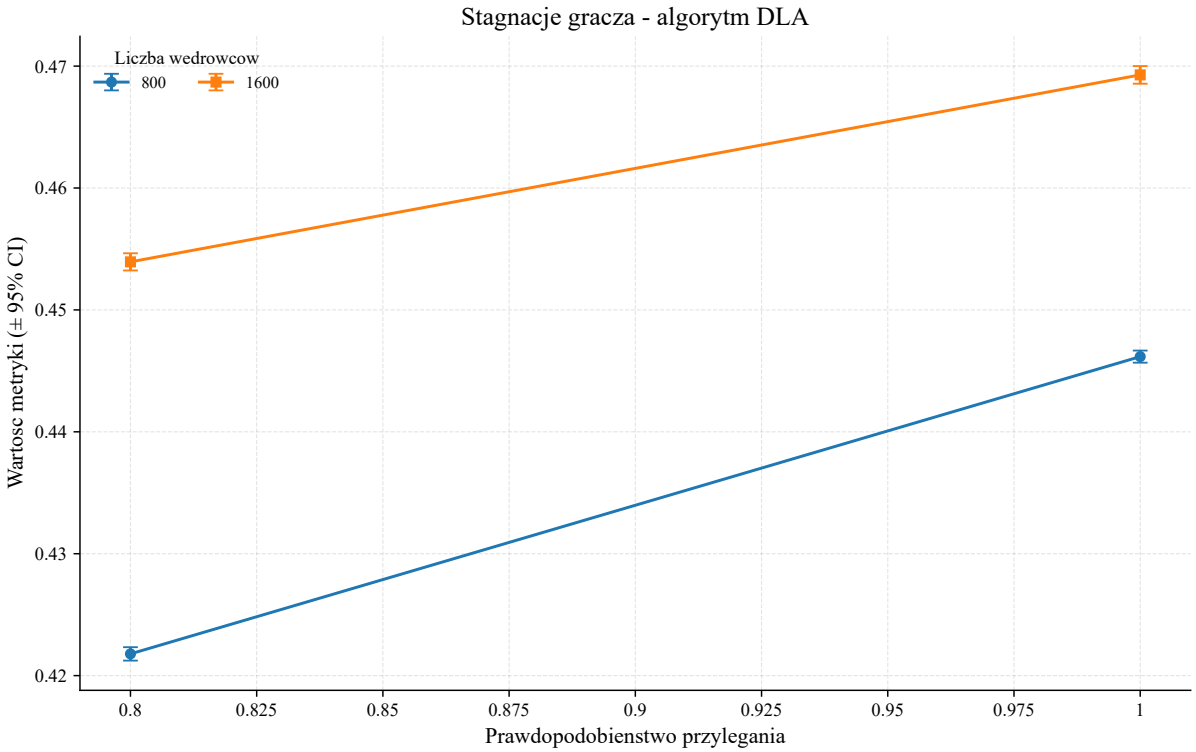
Rysunek 4.18: Relatywne kroki do wyjścia w funkcji liczby wędrowców dla różnych poziomów dryfu do centrum



Rysunek 4.19: Stosunek liniowych korytarzy w funkcji prawdopodobieństwa przylegania dla różnej liczby wędrowców



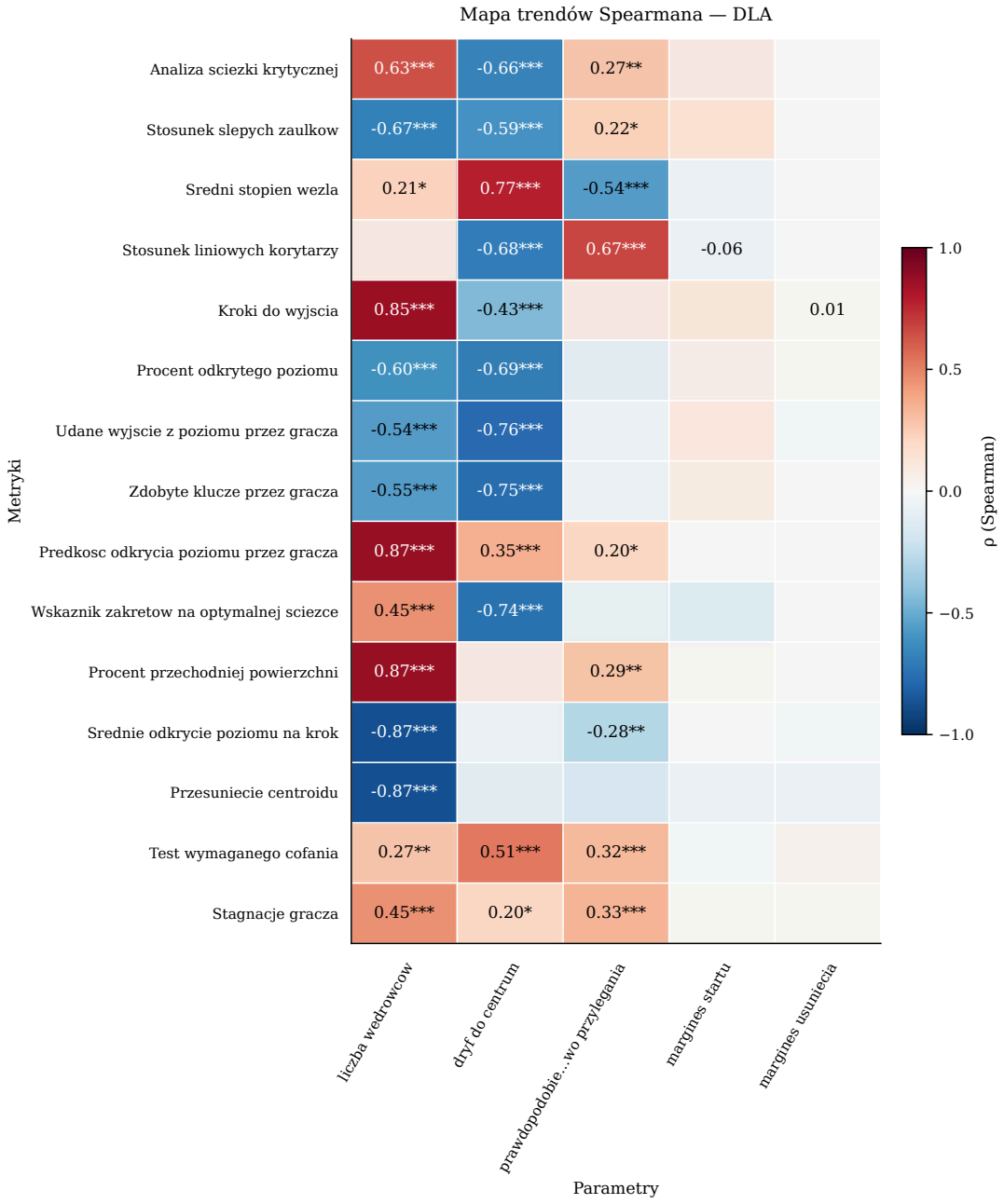
Rysunek 4.20: Średni stopień węzła w funkcji prawdopodobieństwa przylegania dla różnej liczby wędrowców



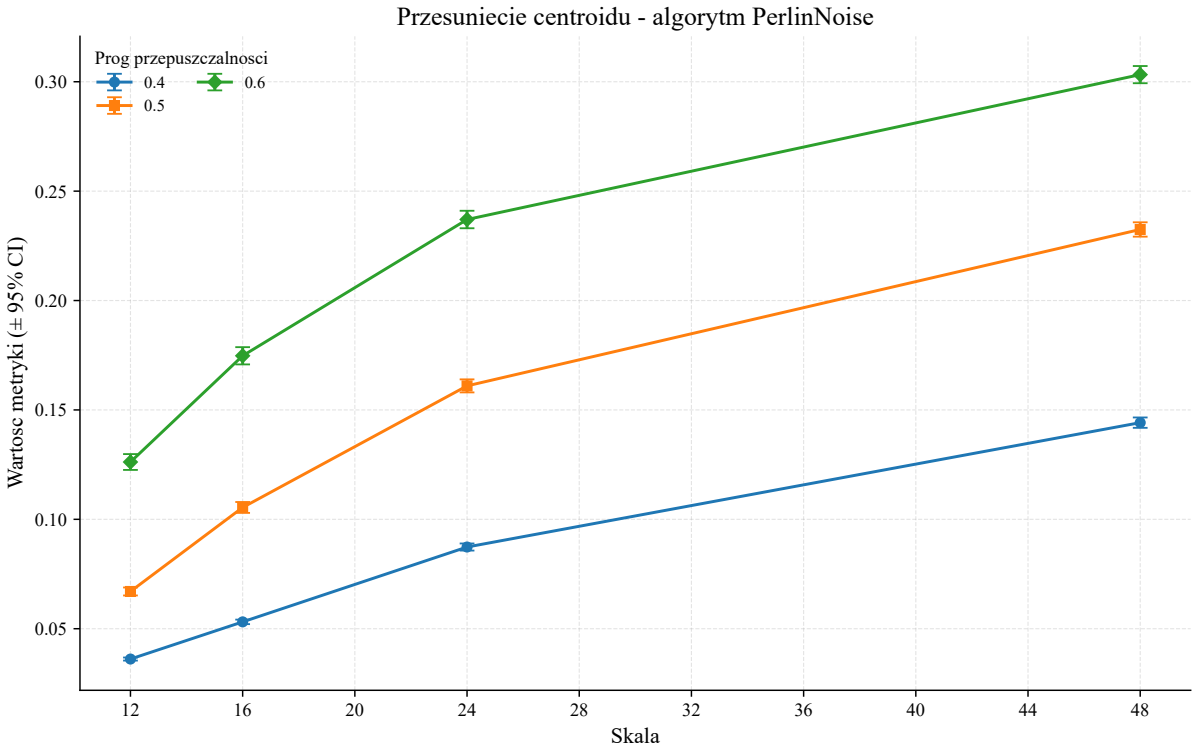
Rysunek 4.21: Stagnacje gracza w funkcji prawdopodobieństwa przylegania dla różnej liczby wędrowców

serwuje się dla liczby wędrowców, która wyraźnie zwiększa przechodniość mapy ($\rho \approx 0{,}87$) i średni stopień węzła, a jednocześnie redukuje ślepe zaułki ($\rho \approx -0{,}67$); dzieje się to jednak kosztem dłuższej trasy do wyjścia ($\rho \approx 0{,}85$) i częstszych stagnacji gracza ($\rho \approx 0{,}45$). Z kolei parametr dryfu do centrum poprawia spójność układu (średni stopień węzła, $\rho \approx 0{,}77$) i redukuje liczbę zakrętów ($\rho \approx -0{,}74$), ale obniża procent odkrytego poziomu ($\rho \approx -0{,}69$) oraz skuteczność w realizacji zadań (wyjścia, klucze). Prawdopodobieństwo przylegania wzmacnia liniowość korytarzy ($\rho \approx 0{,}67$) i sprzyja tunelowości, przy umiarkowanym wzroście cofnięć ($\rho \approx 0{,}32$). Natomiast parametry marginesowe nie wykazują istotnego wpływu ($|\rho| < 0{,}1$), ograniczając się głównie do subtelnych zmian geometrii początkowej.

Wnioski cząstkowe. Algorytm *DLA* generuje poziomy o wysokiej przechodniości, jednak odbywa się to kosztem wydłużenia tras i mniejszego stopnia odkrycia mapy. Najlepszy kompromis można osiągnąć przy umiarkowanej liczbie wędrowców i niewielkim dryfie do centrum, co pozwala zachować równowagę między spójnością a grywalnością. Z kolei wysokie wartości przylegania należy stosować ostrożnie, gdyż prowadzą one do większej liniowości i liczniejszych ślepych zaułków. Algorytm szczególnie dobrze sprawdza się przy tworzeniu poziomów o labiryntowym charakterze, które stanowią dla gracza wymagające wyzwania eksploracyjne.



Rysunek 4.22: Korelacje rangowe Spearmana między metrykami a parametrami algorytmu *DLA* (komórki: ρ). Istotność: * $p < 0{,}05$; ** $p < 0{,}01$; *** $p < 0{,}001$. Adnotacje ograniczono do relacji istotnych ($p < 0{,}05$) lub o sile $|\rho| \geq 0{,}10$.



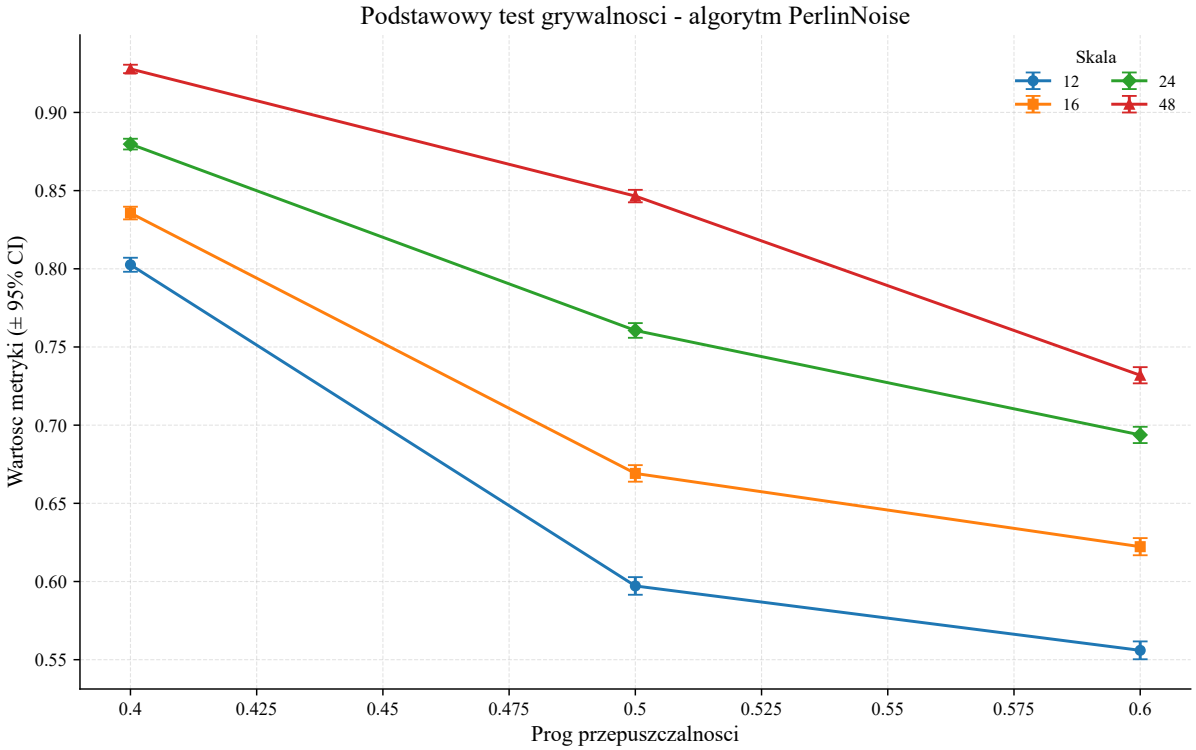
Rysunek 4.23: Przesunięcie centroidu w zależności od skali i progu przepuszczalności dla algorytmu *Szumu Perlina*.

4.6 Algorytm *Szumu Perlina*, PerlinNoise – studium przypadku

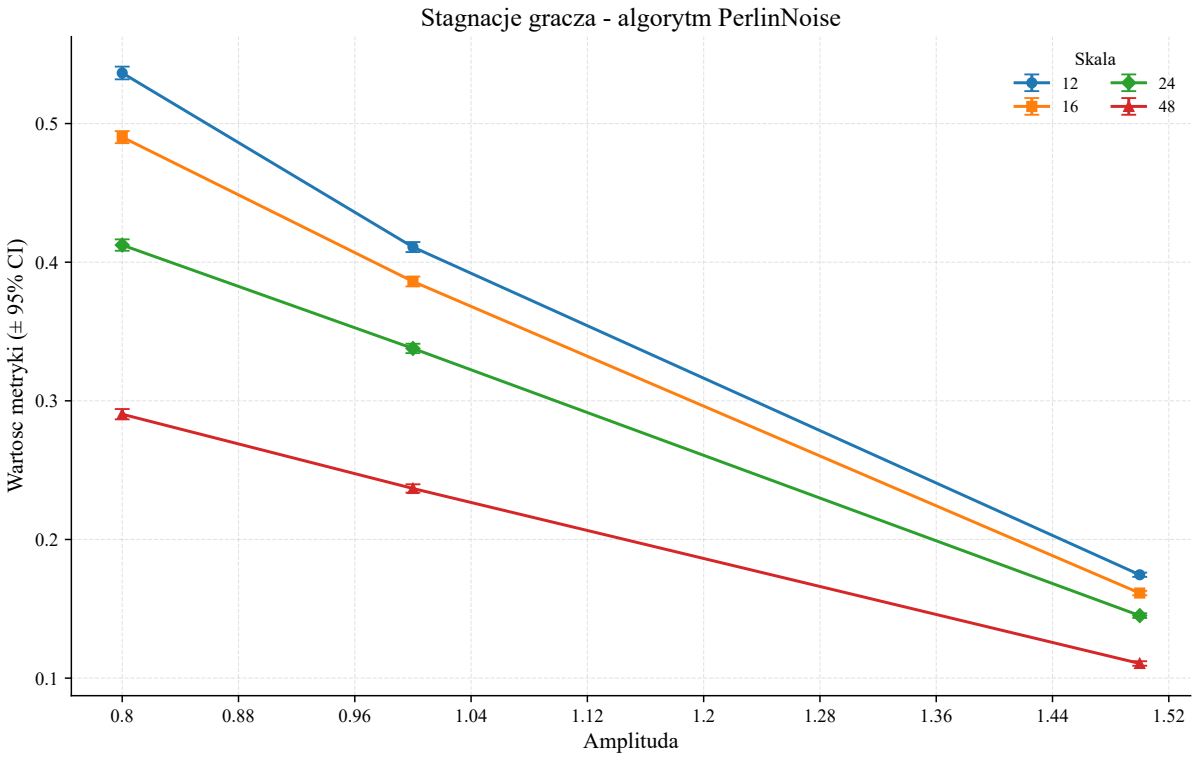
Algorytm *Szumu Perlina* został poddany analizie w różnych konfiguracjach, obejmujących odmienne wartości *progu przepuszczalności* oraz *skali* i *amplitudy*. Wyniki poddano analizie korelacji Spearmana, a istotne wartości uwzględniono przy poziomie $p < 0{,}05$. Łącznie przeanalizowano 144 konfiguracji, co odpowiada około $8{,}64 \times 10^6$ pojedynczym przebiegom symulacji gracza. Plik *data_PerlinNoise.json* z wynikami eksperymentów dołączono do pracy w załączniku.

Wyniki metryk strukturalnych

Średni stopień węzła utrzymywał się na wysokim poziomie – dla konfiguracji o skali 16 osiągał nawet 0.93. Podobnie stabilnie zachowywał się stosunek ślepych zaułków, którego wartości były niskie (0.0017–0.0088), co wskazuje na ich ograniczoną liczbę w strukturze mapy. Przesunięcie centroidu rosło wraz ze wzrostem progu przepuszczalności: od wartości 0.14 przy $p = 0.4$ do ponad 0.53 przy $p = 0.6$ i skali 48 (rys. 4.23).



Rysunek 4.24: Wyniki podstawowego testu grywalności w zależności od progu przepuszczalności i skali dla algorytmu *Szumu Perlina*.



Rysunek 4.25: Stagnacje gracza w zależności od amplitudy i skali dla algorytmu *Szumu Perlina*.

Wyniki metryk grywalności

Testy wykazały duże zróżnicowanie pod względem grywalności. W *podstawowym teście grywalności* osiągano średnio od 0.55 do 0.94 (próg 0.4, skala 48) (rys. 4.24). *Analiza ścieżki krytycznej* ujawniła niewielkie wartości – zwykle poniżej 0.02, co oznacza stosunkowo krótką i przewidywalną ścieżkę optymalną. *Liczba stagnacji gracza* zmniejszała się wraz ze wzrostem amplitudy (rys. 4.25).

Analiza trendów

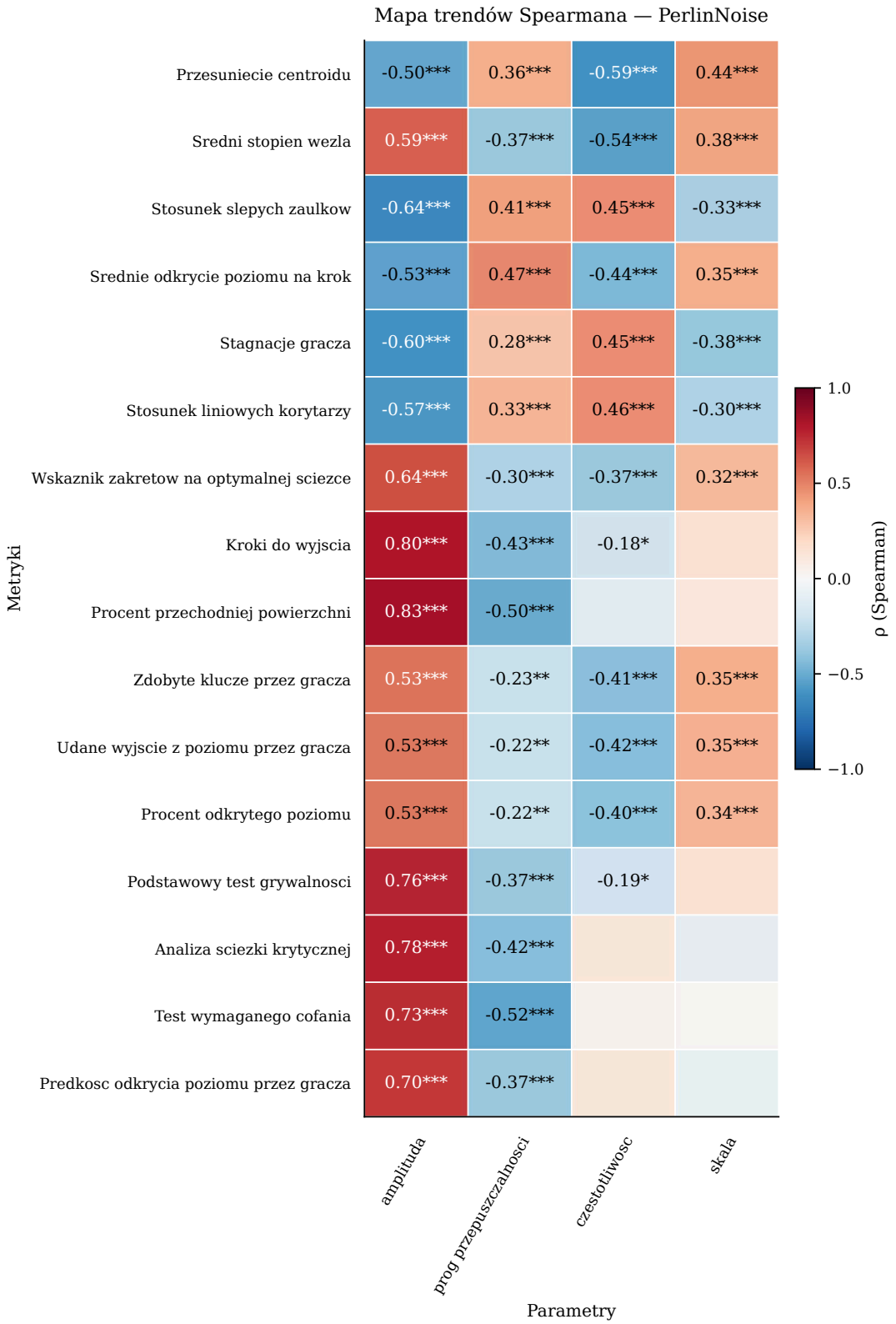
Korelacje Spearmana (rysunek 4.26) wskazują na silne zależności pomiędzy parametrami a metrykami. Amplituda koreluje dodatnio z długością ścieżki (kroki do wyjścia, $\rho = 0.80$) oraz odsetkiem przechodniej powierzchni ($\rho = 0.83$). Wzrost progu przepuszczalności działa w przeciwnym kierunku – osłabia testy grywalności ($\rho = -0.37$) i zwiększa liczbę stagnacji ($\rho = 0.28$). Skala wpływa dodatnio na sukces gracza ($\rho = 0.35$), choć jednocześnie zmniejsza udział ślepych zaułków ($\rho = -0.33$). Częstotliwość, jako parametr kontrolujący ziarnistość szumu, ma umiarkowane znaczenie, równoważąc złożoność trasy i liczbę zaułków.

Wnioski cząstkowe

Analiza algorytmu *Szumu Perlina* pokazuje, że jest to metoda zdolna do generowania poziomów zarówno prostych i czytelnych, jak i bardziej złożonych i wymagających. Przy niskim progu przepuszczalności i mniejszych skalach struktura pozostaje przejrzysta, a trasy krótsze, co skutkuje większą płynnością eksploracji i wysokim odsetkiem sukcesu gracza. Z kolei przy wyższych progach i dużych skalach mapy stają się trudniejsze: rosną długości ścieżek i wskaźniki stagnacji, a tempo odkrywania maleje. Jednocześnie jednak zwiększa się przechodniość oraz spójność układu, co sprzyja bardziej immersyjnej, „labiryntowej" eksploracji. Ostatecznie to właśnie balans pomiędzy skalą a progiem przepuszczalności decyduje o charakterze poziomu: od szybkich ścieżek po rozbudowane układy wymagające cierpliwego odkrywania.

4.7 Algorytm *Prosty Pokój*, SimpleRoom – studium przypadku

Algorytm *Prosty Pokój* opiera się na generowaniu poziomów w postaci prostych, regularnych pomieszczeń w kształcie kwadratu. W badaniach analizowano dwa parametry: *udział punktów* oraz *długość boku kwadratu*. Wyniki poddano analizie korelacji Spearmana, a istotne wartości uwzględniono przy poziomie $p < 0{,}05$. Łącznie przeanalizowano 25 kon-



Rysunek 4.26: Korelacje rangowe Spearmana między metrykami a parametrami algorytmu *Szumu Perlina* (komórki: ρ). Istotność: * $p < 0{,}05$; ** $p < 0{,}01$; *** $p < 0{,}001$. Adnotacje ograniczono do relacji istotnych ($p < 0{,}05$) lub o sile $|\rho| \geq 0{,}10$.

figuracji, co odpowiada około $1{,}5 \times 10^6$ pojedynczym przebiegom symulacji gracza. Plik *data_SimpleRoom.json* z wynikami eksperymentów dołączono do pracy w załączniku.

Analiza wpływu parametrów na metryki

Zależności zestawiono na rys. 4.27. Najsilniejszy wpływ wywiera długość boku kwadratu: wraz ze wzrostem rozmiaru rośnie zarówno powierzchnia przechodnia, jak i liniowość korytarzy, natomiast wyraźnie spadają metryki realizacji celu. Udział punktów (gęstość przeszkód) oddziałuje w kierunku przeciwnym: skraca ścieżki i przyspiesza odkrywanie poziomu, lecz nie usuwa zjawiska stagnacji ani tendencji do struktury monotonicznej. W praktyce większe układy generują rozległe, słabo zróżnicowane przestrzenie, podczas gdy mniejsze sprzyjają układom zwartym i bardziej wymagającym, ale zarazem mniej „otwartym".

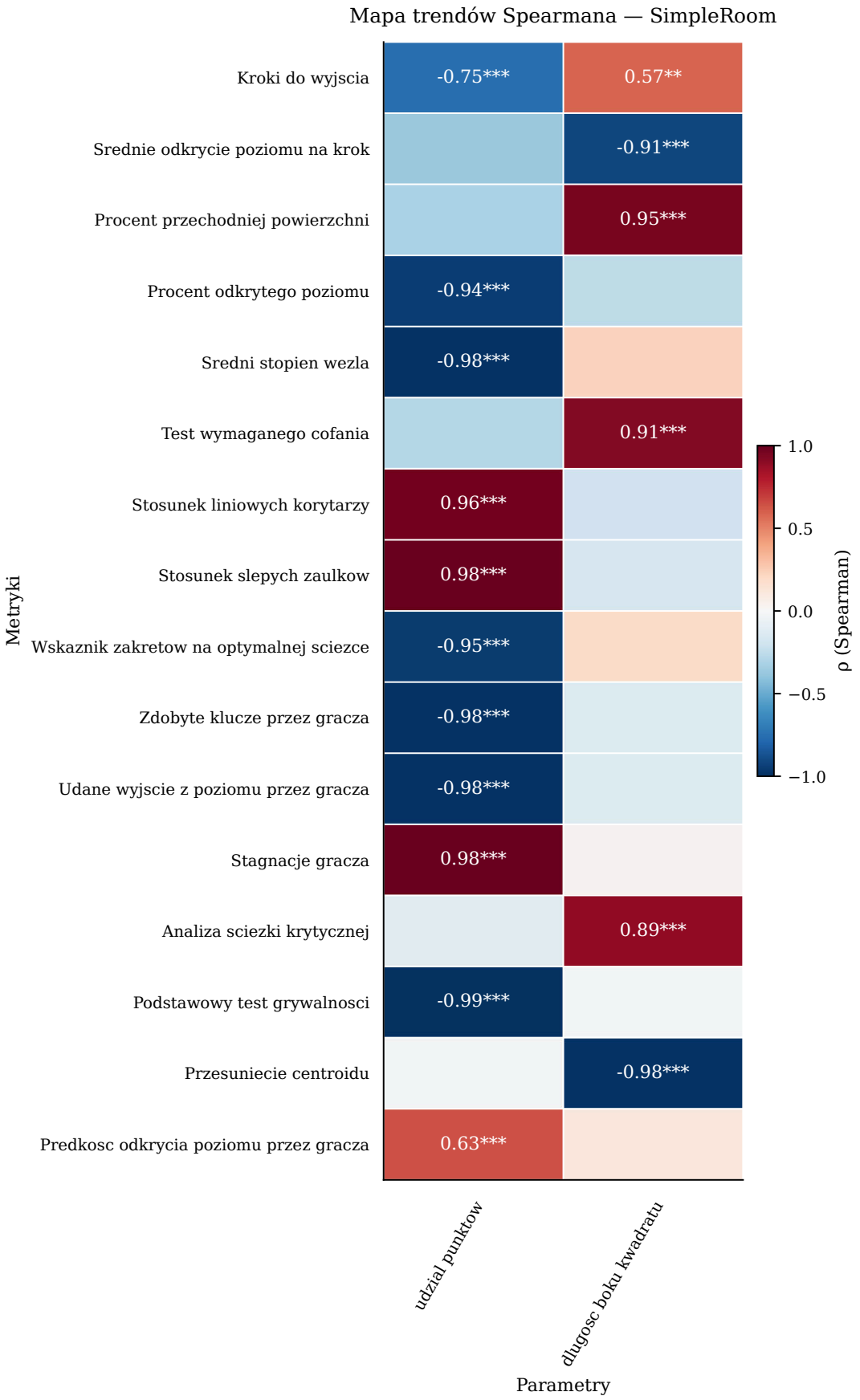
Powierzchnia przechodnia i struktura mapy

Wyniki wskazują na silny związek parametrów z odsetkiem powierzchni przechodniej. Długość boku kwadratu jest dodatnio skorelowana z przechodniością ($\rho \approx 0{,}95$), co oznacza, że większe układy oferują bardziej otwarte przestrzenie. Jednocześnie obserwujemy ujemną korelację z procentem odkrytego poziomu ($\rho \approx -0{,}94$), tzn. gracz eksploruje relatywnie mniejszą część generowanego obszaru. Dla mniejszych wartości parametrów struktura staje się bardziej spójna i gęsta, lecz trudniejsza do pełnego przejścia. Udział punktów działa w kierunku przeciwnym: zwiększa procent odkrycia ($\rho \approx 0{,}63$), ale obniża przechodniość ($\rho \approx -0{,}76$). Większa liczba przeszkód prowadzi więc do bardziej złożonej, labiryntowej struktury, wymagającej dokładniejszej eksploracji w poszukiwaniu wyjścia.

Grywalność i zachowanie gracza

Pod względem grywalności odnotowujemy silne, ujemne korelacje z długością boku kwadratu: liczba udanych wyjść ($\rho \approx -0{,}98$) oraz zdobytych kluczy ($\rho \approx -0{,}98$) spadają dla dużych wymiarów, a częstość stagnacji gracza rośnie ($\rho \approx 0{,}98$). Oznacza to większą podatność na utknięcie w rozległych fragmentach poziomu. Z kolei wyższy udział punktów poprawia dynamikę rozgrywki, zwiększając tempo odkrywania ($\rho \approx 0{,}63$), nie kompensuje jednak negatywnego wpływu rozmiaru mapy na osiąganie celów.

Wnioski cząstkowe. Algorytm *Prosty Pokój* generuje układy o silnie deterministycznej strukturze, w której równowaga między wielkością a grywalnością jest szczególnie trudna do osiągnięcia. Większe wartości parametru *długość boku* poprawiają przechodniość i prostotę tras, ale jednocześnie wprowadzają wysoką tunelowość, liczne ślepe zaułki oraz spadek sukcesów gracza. Z kolei zwiększanie udziału punktów ułatwia eksplorację i skraca ścieżkę krytyczną, lecz nie usuwa ograniczeń wynikających z samej geometrii algorytmu.



Rysunek 4.27: Korelacje rangowe Spearmana między metrykami a parametrami algorytmu *Prosty Pokój* (komórki: ρ). Istotność: * $p < 0{,}05$; ** $p < 0{,}01$; *** $p < 0{,}001$. Adnotacje ograniczono do relacji istotnych ($p < 0{,}05$) lub o sile $|\rho| \geq 0{,}10$.

Ostatecznie *Prosty Pokój* może być metodą metodą odpowiednią do projektowania poziomów o klarownej, prostoliniowej strukturze, lecz mniej atrakcyjną w kontekście złożonej eksploracji i zbalansowanej grywalności.

4.8 Poglądowa analiza porównawcza algorytmów

W tym rozdziale ograniczono się do metryk, które najlepiej różnicują badane algorytmy i jednocześnie są najbardziej użyteczne interpretacyjnie z perspektywy projektowania poziomów. Przed rozpoczęciem analiz odrzucamy poziomy którym nie udało się przejść *podstawowego testu grywalności*, tak aby porównania dotyczyły wyłącznie konfiguracji spełniających minimalne kryterium jakości. Dla każdej z poniższych miar przedstawiamy dwa ujęcia:

- Rozkłady wartości dla wszystkich algorytmów, zagregowane w poprzek konfiguracji.
- Średnie wartości dla każdego algorytmu, w jednolitym i stałym porządku we wszystkich wykresach.

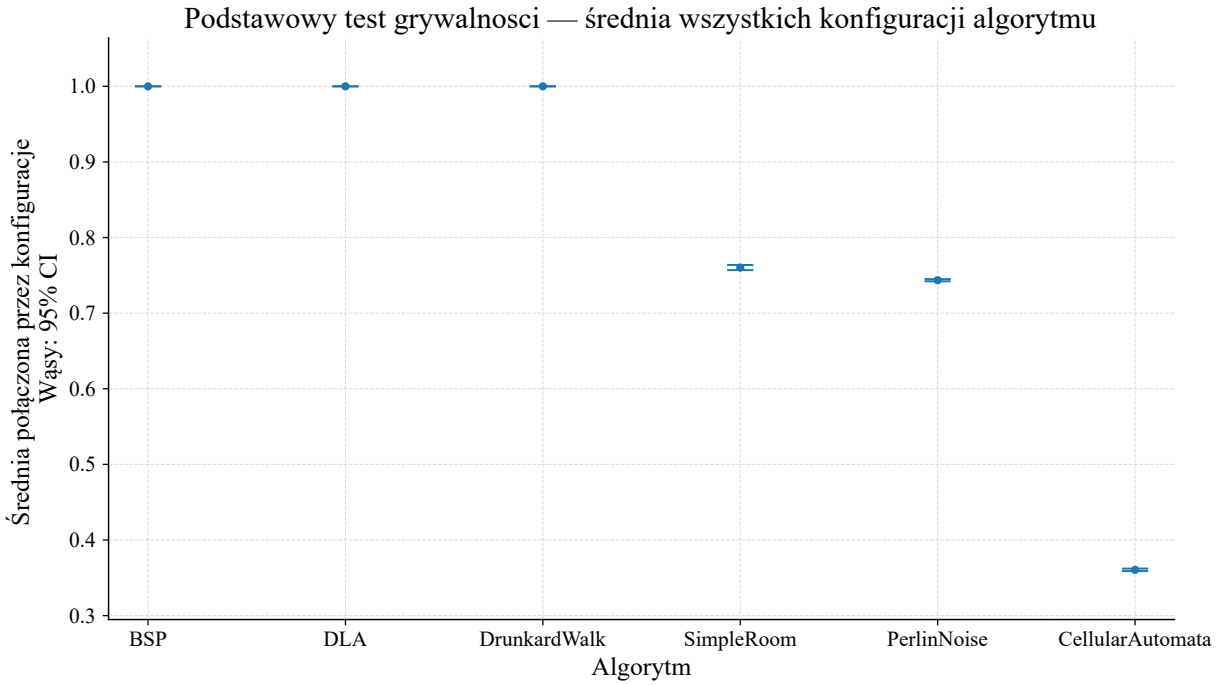
Położenie rozkładu oraz wartość średnia wskazują, gdzie w danej mierze lokuje się wynik typowy algorytmu. Szerokość rozkładu, rozumiana jako rozproszenie wyników, stanowi poglądową miarę stabilności na tle wszystkich konfiguracji. Kształt rozkładu informuje o skłonności do wyników skrajnych lub o istnieniu odmiennych klas wyników.

4.8.1 Podstawowy test grywalności — charakterystyka

Podstawowy test grywalności to najprostsza z badanych miar, określana jako odsetek instancji poziomów, które spełniają minimalne warunki grywalności. Wymagania te obejmują istnienie spójnej ścieżki między punktami startu i wyjścia oraz obecność wystarczającej ilości przestrzeni przechodniej. Wskaźnik ten jest obliczany dla każdej konfiguracji algorytmu jako stosunek liczby wygenerowanych poziomów spełniających te kryteria do całkowitej liczby wygenerowanych poziomów. Wskaźnik podstawowego testu grywalności interpretujemy jako odsetek instancji wygenerowanych przez dany algorytm, które spełniają minimalne warunki grywalności. Na rysunku 4.28 przedstawiono średnie tego wskaźnika obliczone w poprzek konfiguracji, wraz z 95% przedziałami ufności wyznaczonymi na podstawie błędu standardowego średniej. Rysunek 4.29 prezentuje odpowiadające im rozkłady wartości dla poszczególnych algorytmów.

Wyniki zbiorcze i ich interpretacja.

Wyniki dzielą badane metody na trzy wyraźne grupy. *BSP*, *DLA* i *Pijany Spacer* osiągają średnie bliskie jedności, a ich rozkłady są silnie skupione przy wartości maksymalnej.



Rysunek 4.28: Średnie wyniki podstawowego testu grywalności dla badanych algorytmów. Wąsy przedstawiają 95% przedziały ufności.

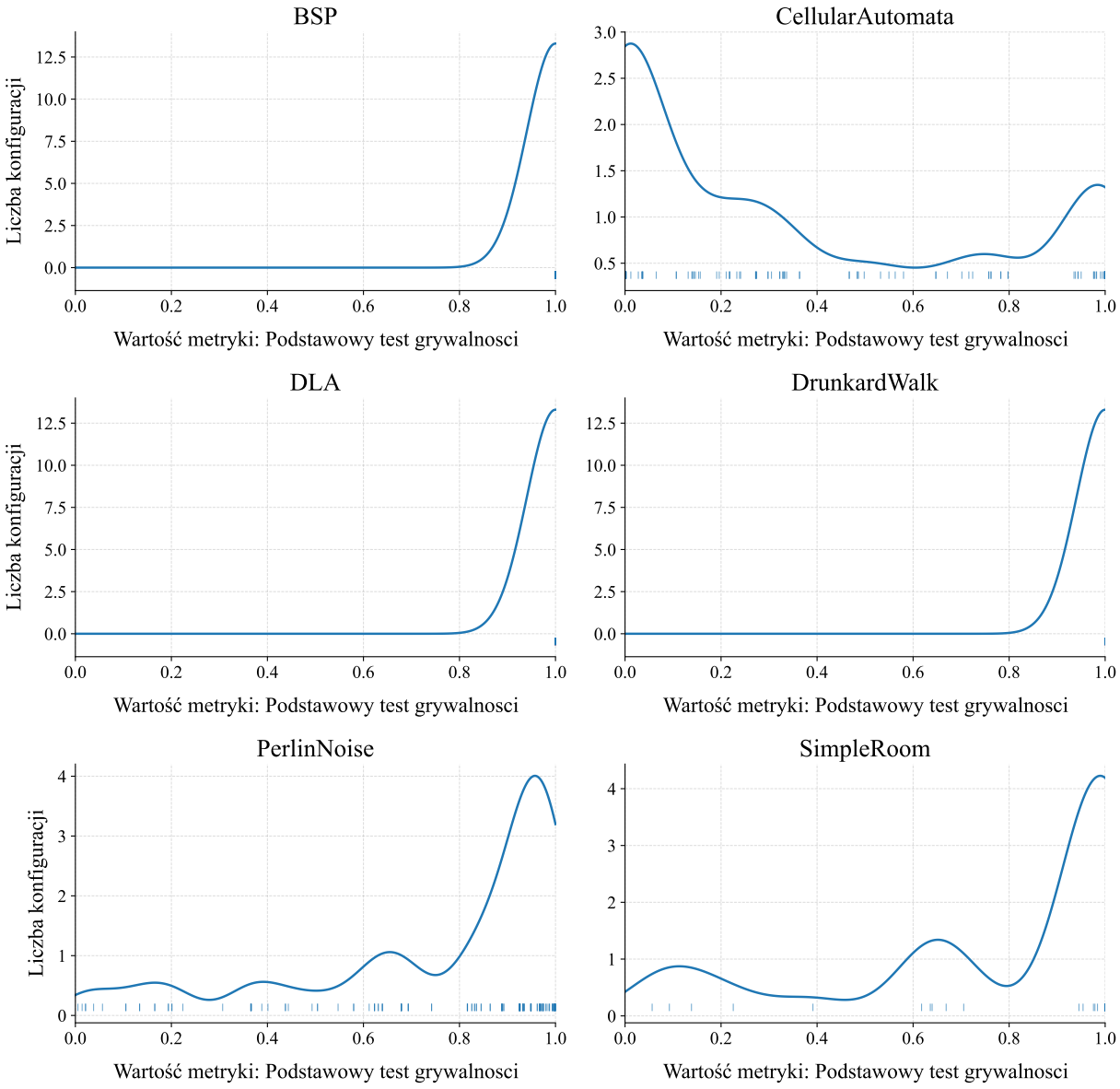
Oznacza to, że niemal każda konfiguracja tych algorytmów spełnia próg grywalności. Druga grupa obejmuje *Prosty Pokój* oraz *Szum Perlina*. Dla obu metod średnia wartość wskaźnika jest wysoka (około 0,75), a rozkłady mają charakter wyraźnie wielomodalny. Wskazuje to na istnienie odmiennych schematów działania zależnych od ustawień parametrów sprawiające, że część z konfiguracji generuje niegrywalne poziomy. Trzeci przypadek stanową *Automaty Komórkowe*, dla których średnia jest najniższa (około 0,35), a rozkład najszerszy i również wielomodalny. Wynik ten sugeruje, że bez dodatkowych etapów "naprawiających"poziom znaczna część konfiguracji nie spełnia minimalnego warunku grywalności.

4.8.2 Analiza ścieżki krytycznej — charakterystyka

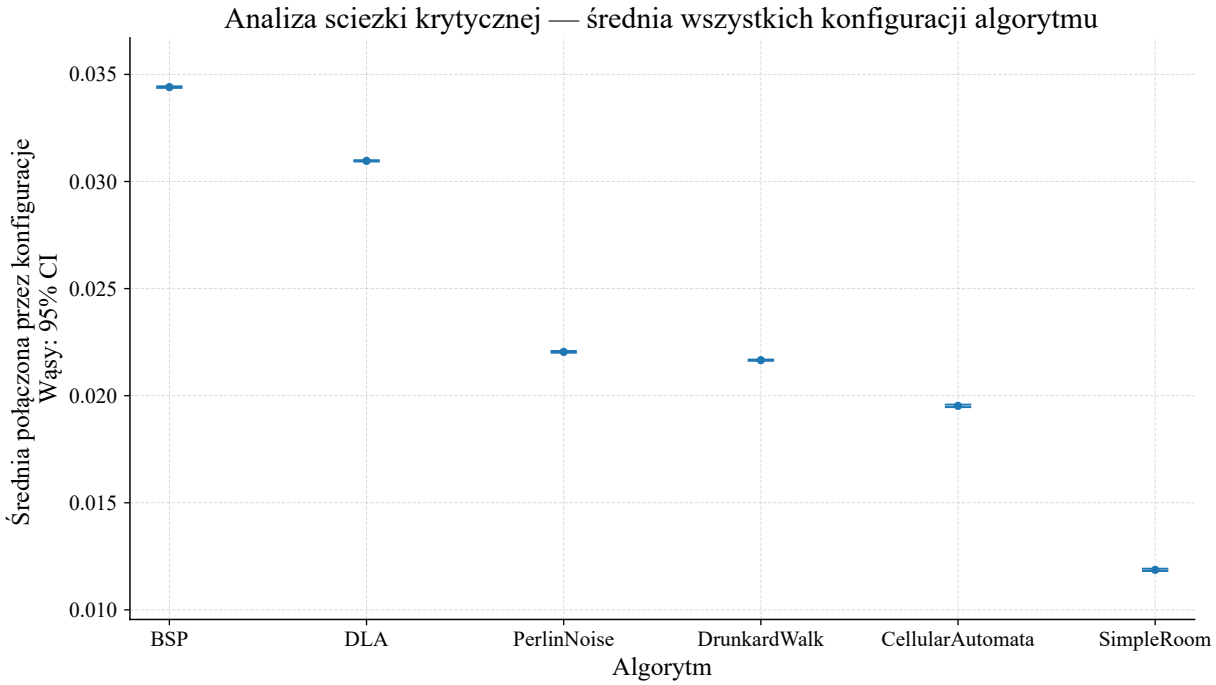
Miara „analiza ścieżki krytycznej” opisuje względną długość obowiązkowego przebiegu od pola startowego przez pole klucza do wyjścia ($S \to K \to E$) i normalizuje przez liczbę pól planszy, tak aby porównanie było niezależne od rozmiaru siatki – niższe wartości oznaczają układy bardziej zwarte.

Wyniki zbiorcze i ich interpretacja.

Porządek średnich (rys. 4.30) wskazuje na wyraźne różnice między podejściami. *Prosty Pokój* uzyskuje najkrótszą ścieżkę obowiązkową rozkład gęstości jest silnie skoncentrowany w dolnym zakresie wartości z niewielkim drugim wierzchołkiem, co sugeruje, że jedno-pomieszczeniowa geometria, z natury minimalizująca odległości między S, K i E,



Rysunek 4.29: Jądrowe estymacje gęstości *podstawowego testu grywalności* dla każdego algorytmu.



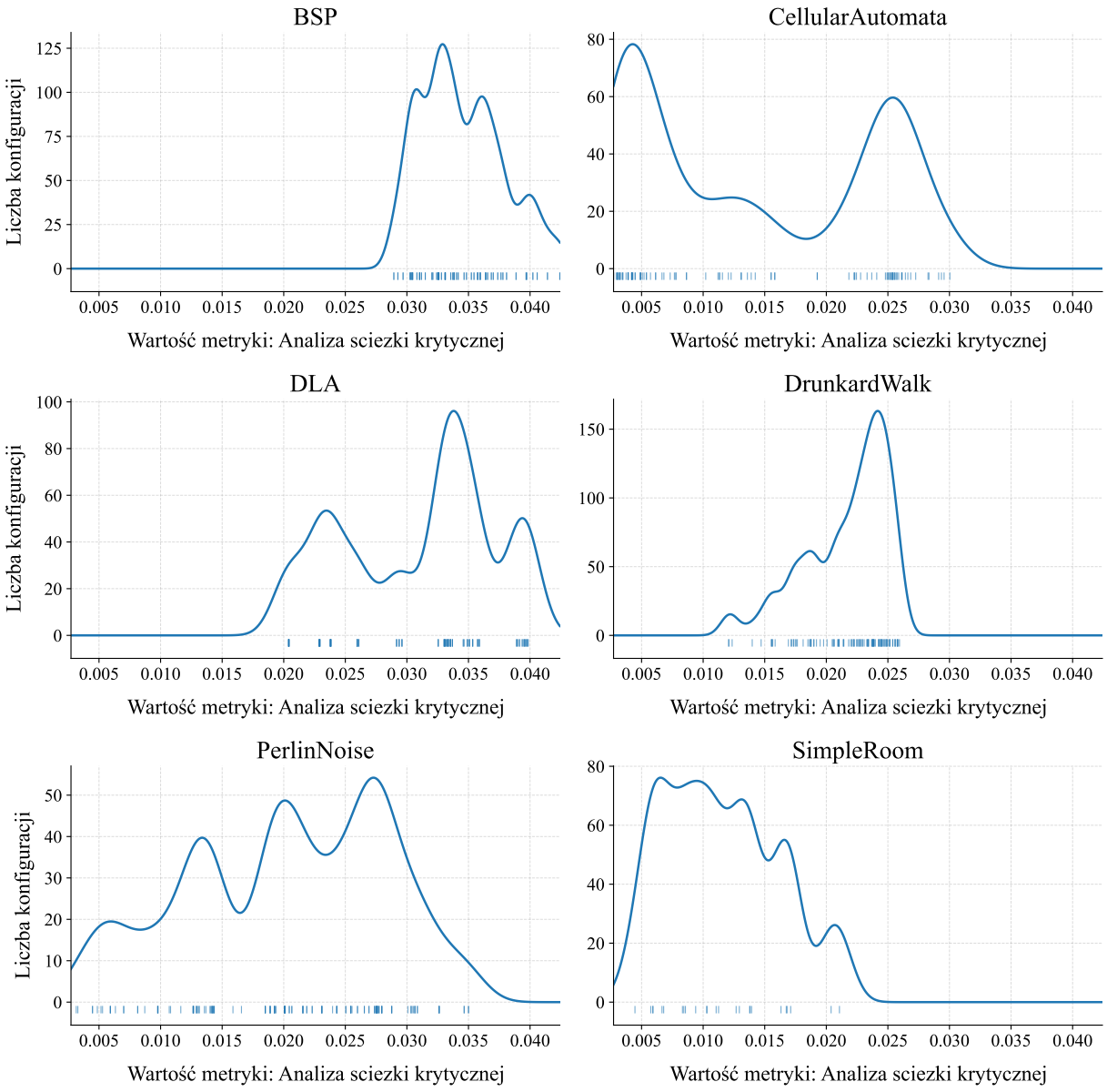
Rysunek 4.30: Średnie wyniki analizy ścieżki krytycznej ($S \to K \to E$) dla badanych algorytmów. Wąsy przedstawiają 95% przedziały ufności.

zapewnia zwarty przebieg przy umiarkowanej zależności od ustawień. Rozkład *Automatów Komórkowych* (rys. 4.31) jest szeroki i wielomodalny co sugeruje, że obok konfiguracji dających krótkie przejścia pojawiają się warianty generujące układy bardziej rozproszone. W przypadku *Pijanego Spaceru* dominuje wąski wierzchołek w okolicach wartości średniej (miara względnie stabilna), natomiast *Szum Perlina* charakteryzuje się kilkoma wyraźnymi maksimami na osi wartości, co wskazuje na istotną wrażliwość na próg binarizacji i parametry przetwarzania. Najdłuższą ścieżkę obowiązkową uzyskują *BSP* i *DLA*. W *BSP* rozkład jest stosunkowo wąski i przesunięty ku wyższym wartościom, co można wiązać z regularnym podziałem przestrzeni naturalnie wydłużającej trasę $S \to K \to E$. W *DLA* rozkład jest szeroki i wielomodalny, co wskazuje na istnienie różnych reżimów działania zależnych od parametrów, generujących zarówno zwarte, jak i rozproszone układy.

Implikacje dla projektowania i dalszych porównań. W ujęciu poglądowym algorytmy preferujące układy zwarte sprzyjają krótkiej ścieżce obowiązkowej, natomiast metody budujące rozbudowane ciągi korytarzy i rozdzielające strefy funkcjonalne prowadzą do dłuższych przejść.

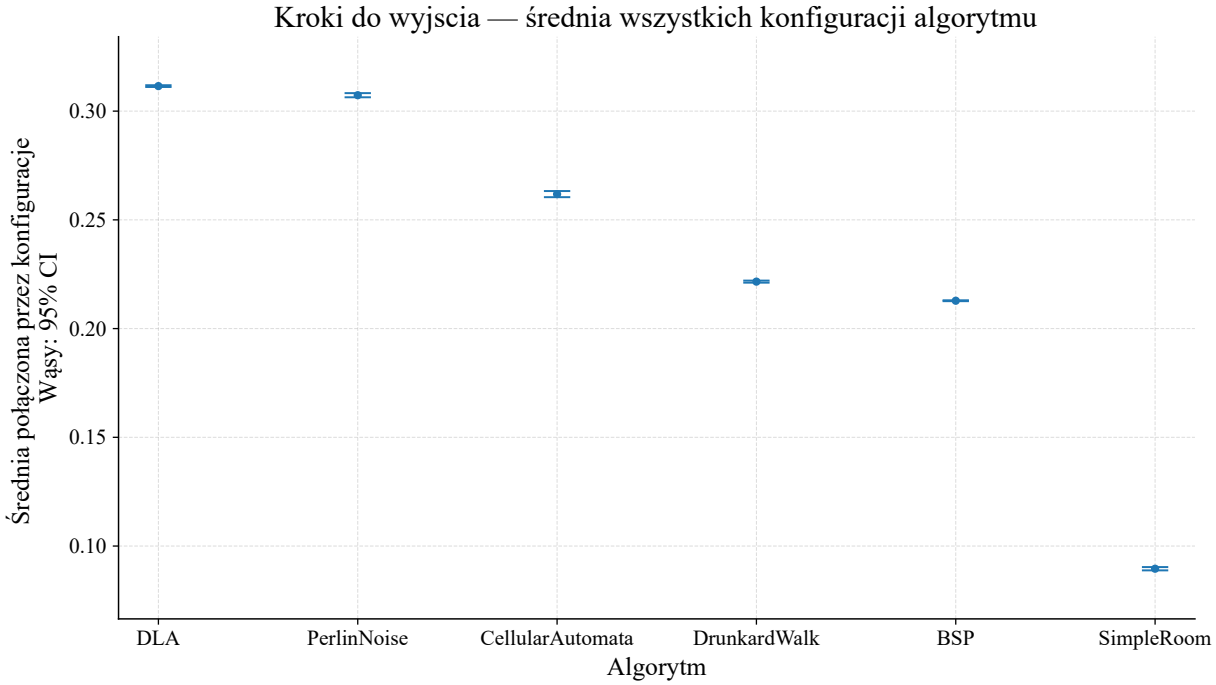
4.8.3 Kroki do wyjścia — charakterystyka

Metryka „kroki do wyjścia” mierzy liczbę kroków potrzebnych, jaką wirtualny gracz po zdobyciu klucza dotarł do wyjścia ($S \to K \to E$), znormalizowaną przez liczbę pól planszy. Do uśredniania wchodzą wyłącznie udane przebiegi symulacji. Niższe wartości



Rysunek 4.31: Jądrowe estymacje gęstości *analizy ścieżki krytycznej* dla każdego algorytmu.

oznaczają bardziej kompaktowe układy i krótszą drogę do wyjścia, wyższe — rozległe, „dłuższe” w przejściu lochy. Wynik odzwierciedla zachowanie deterministycznego agenta.

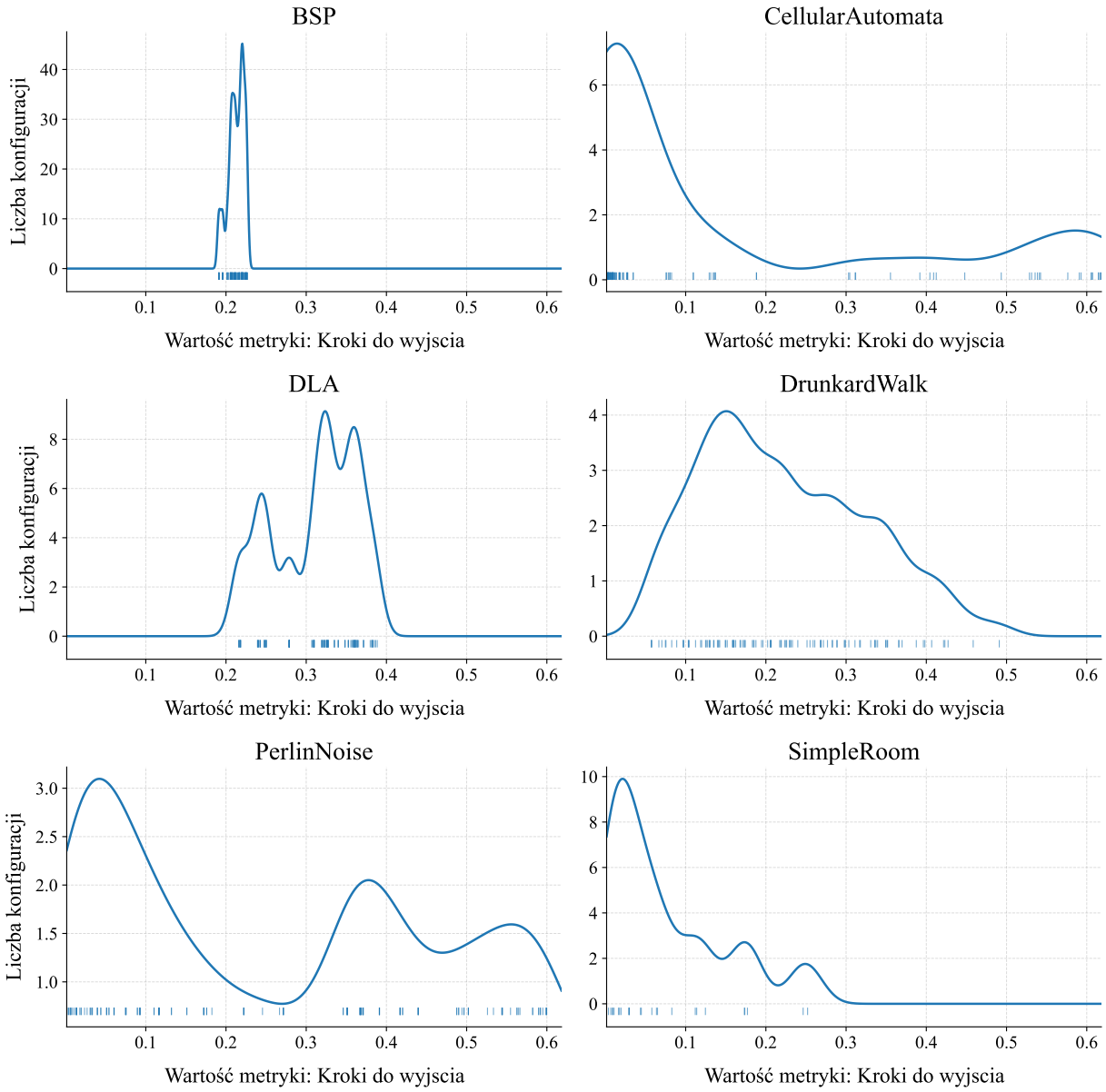


Rysunek 4.32: Średnie, znormalizowane wyniki ilości kroków do wyjścia dla badanych algorytmów. Wąsy przedstawiają 95% przedziały ufności.

Wyniki zbiorcze i ich interpretacja.

Zestawienie średnich (rys. 4.32) porządkuje badane metody według „długości” typowego przejścia. Najkrótszą drogę do wyjścia zapewnia *Proty Pokój*: jednopomieszczeniowa geometria z natury skraca odległości między punktami startu, klucza i wyjścia, co przekłada się na zwięzły przebieg rozgrywki. Nieco dłuższe, lecz wciąż zwarte przejścia obserwujemy w *BSP* i *Pijany Spacer*. W pierwszym przypadku wynika to z regularnego rytmu komnat i korytarzy, w drugim — z powstającej siatki tuneli, która, mimo krętości, nie rozciąga poziomu nadmiernie. *Automaty Komórkowe* lokują się wyżej, sygnalizując większą rozciągłość lub krętość tras. Najdłuższe przejścia występują w *Szumie Perlina* oraz *DLA*.

Rozkłady przedstawione na rys. 4.33 przedstawiają o informację o stabilności. W *Prostym Pokoju* wartości skupiają się w dolnym zakresie, z jedynie śladowym drugim wierzchołkiem, co wskazuje na wysoką powtarzalność osiąganych wyników. *BSP* daje rozkład wąski, a więc odporny na zmiany ustawień. W *Pijanym Spacerze* jądro rozkładu jest szersze, ale wciąż skoncentrowane wokół wartości typowych. *Automaty Komórkowe* cechuje się rozkładem szerokim, z wyraźnym ogonem po stronie większych wartości, co odzwierciedla współistnienie konfiguracji zwartych i takich, które rozciągają przejście. *DLA* oraz *Szum Perlina* mają rozkłady, świadczące o odmiennych zróżnicowanych długościach trasy do



Rysunek 4.33: Jądrowe estymacje gęstości *kroków do wyjścia* dla każdego algorytmu.

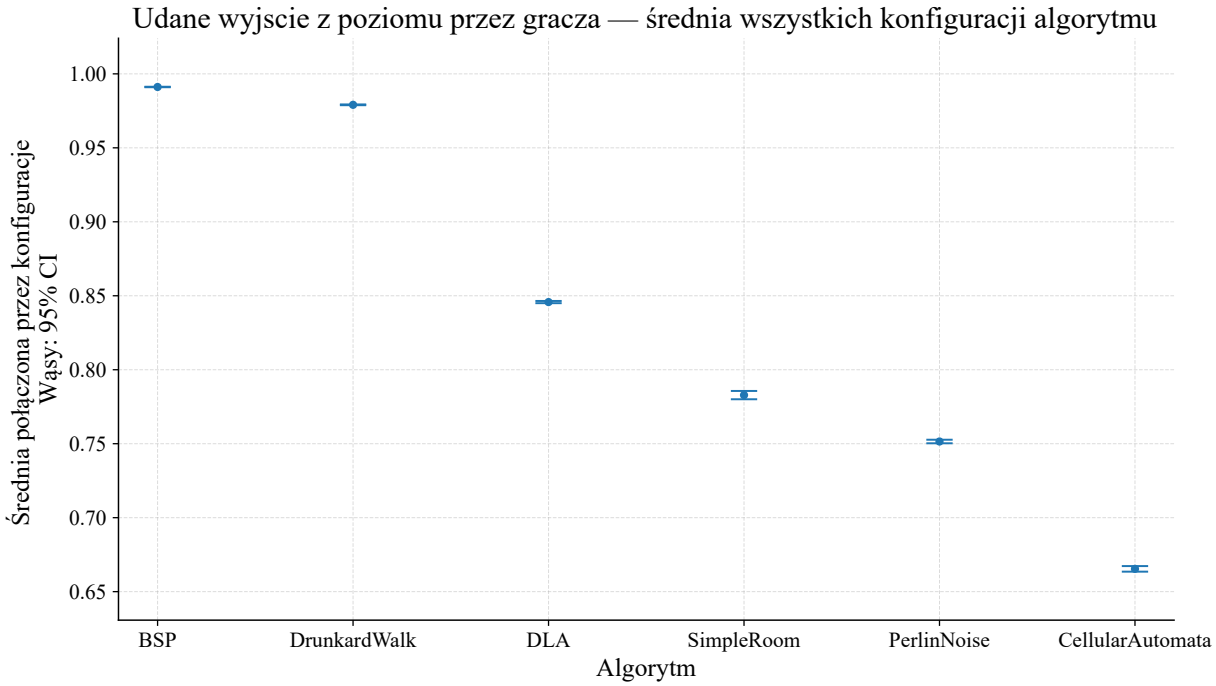
wyjścia.

Relacja do „ścieżki krytycznej" i implikacje projektowe. Porządek metod jest spójny z *Analizą ścieżki krytycznej*: układy zwarte skracają zarówno minimalny przebieg $S \to K \to E$, jak i realne kroki gracza, natomiast układy z korytarzami i podziałami sprzyjają dłuższym przejściom. *Kroki do wyjścia* dodatkowo „karzą" nieefektywną nawigację i zapętlenia agenta, przez co są zwykle większe niż sama ścieżka krytyczna.

4.8.4 Udane wyjścia z poziomu przez gracza — charakterystyka

Wskaźnik „udane wyjścia z poziomu przez gracza" określa odsetek przebiegów, w których wirtualny gracz najpierw zdobywa klucz, a następnie dociera do wyjścia. Formalnie

jest to udział sukcesów w serii niezależnych symulacji, przy czym sukces definiuje się jako spełnienie obu warunków w trakcie pojedynczego przebiegu. Warto pamiętać, że metryka ta wynika z działania deterministycznego agenta który, realizując hierarchię celów w pierwszej kolejności dąży on do klucza, a następnie do wyjścia, a w przeciwnym razie eksploruje najbliższe punkty graniczne.

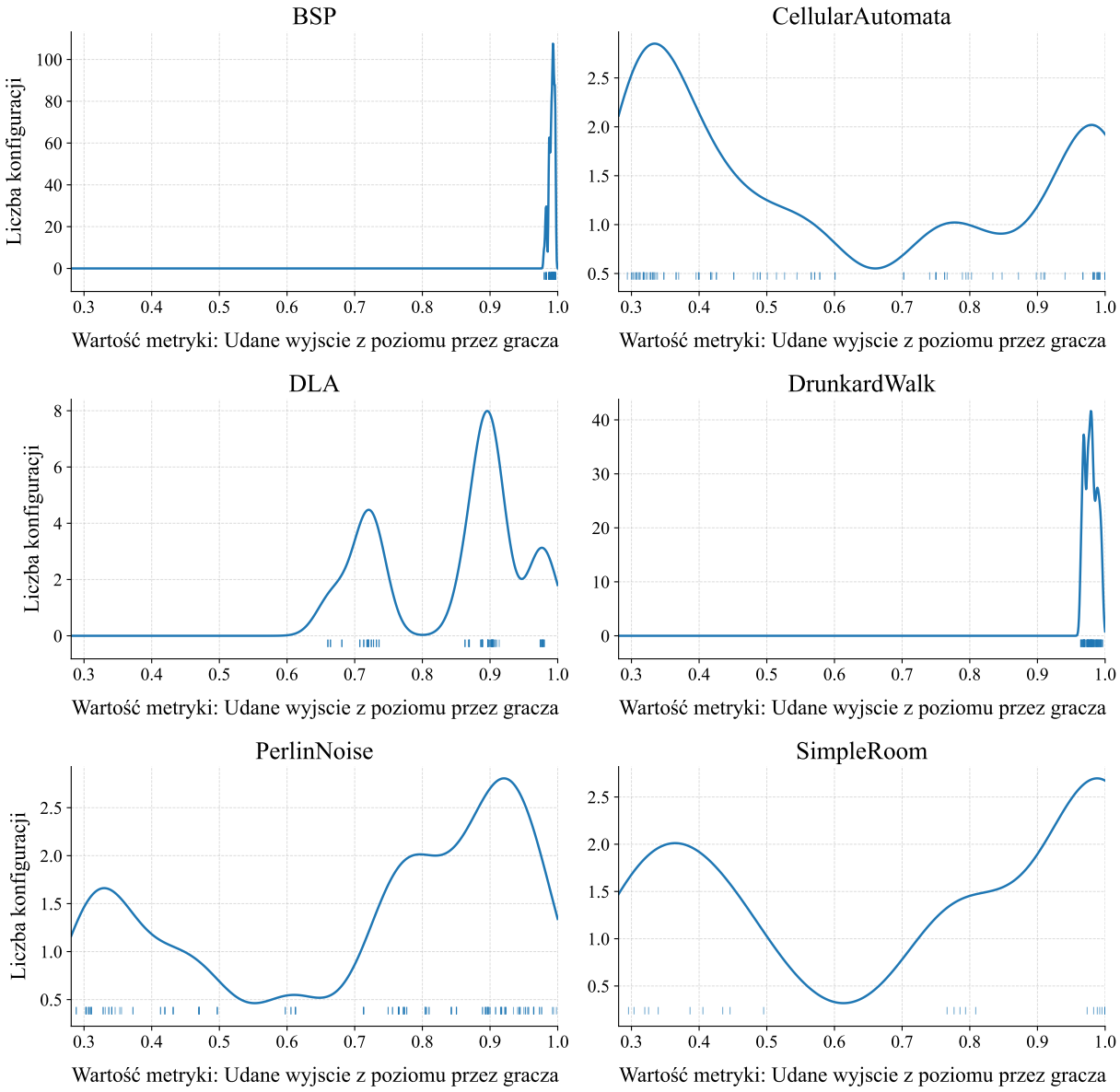


Rysunek 4.34: Średnie wyniki udanych wyjść z poziomu dla badanych algorytmów. Wąsy przedstawiają 95% przedziały ufności.

Wyniki zbiorcze i ich interpretacja.

Na rys. 4.34 możemy zaobserwować, że najwyższy odsetek udanych wyjść uzyskują *BSP* oraz *Pijany Spacer*. W tych algorytmach łączność przestrzeni i czytelna organizacja przejść sprzyjają sprawnej realizacji sekwencji „klucz → wyjście". Nieco niżej plasuje się *DLA*, które — mimo rozbudowanej geometrii — zazwyczaj zapewnia wystarczającą spójność dla skutecznej nawigacji. Kolejna grupa to *Prosty Pokój* oraz *Szum Perlina* których wyniki są tu wyraźnie niższe i bardziej zależne od ustawień. Najniższy poziom skuteczności obserwuje się w *Automatach Komórkowych*, gdzie częściej dochodzi do sytuacji wymagających długotrwałej eksploracji lub generujących zapętlenia ruchu.

Krzywe gęstości (rys. 4.35) potwierdzają te różnice. W *BSP* i w algorytmie *Pijany Spacer* rozkłady są silnie skupione przy wartościach bliskich maksimum, co świadczy o wysokiej stabilności między konfiguracjami. *DLA* ma rozkład szerszy i wielowierzchołkowy, odzwierciedlający współistnienie konfiguracji bardzo skutecznych i takich, które wymagają dłuższej wędrówki, lecz wciąż częściej kończą się powodzeniem. *Szum Perlina* cechuje się wyraźną wielomodalnością oraz dłuższym ogonem, co wiąże się z wrażliwo-

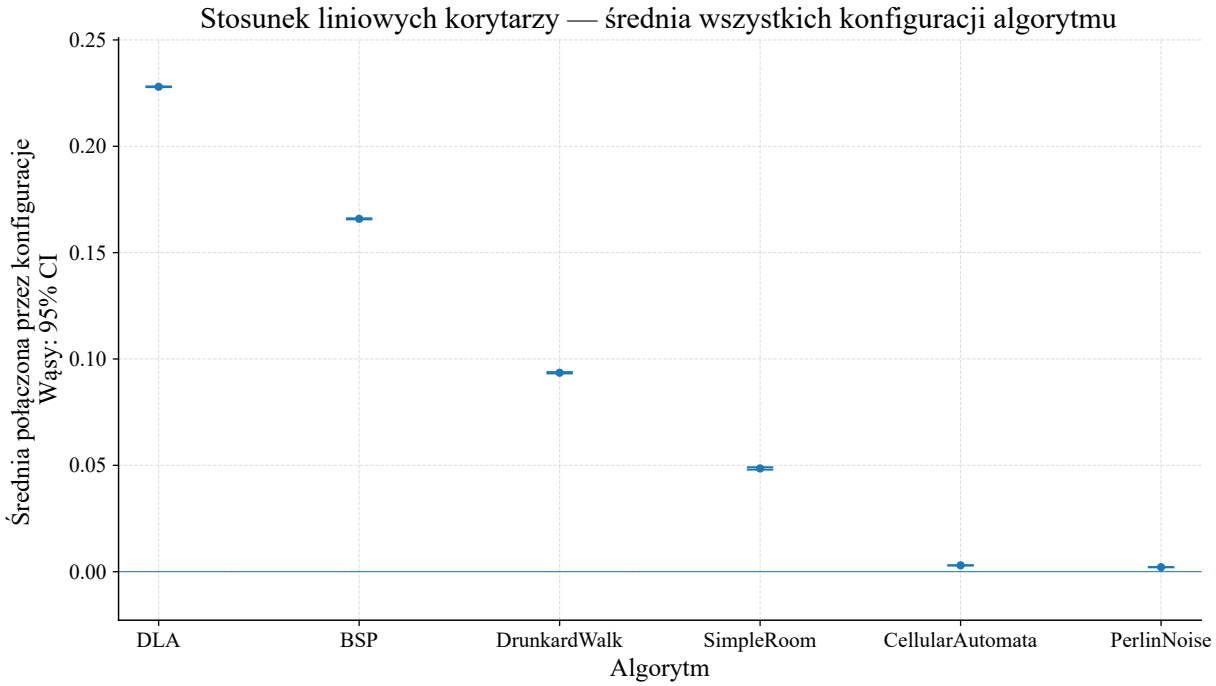


Rysunek 4.35: Jądrowe estymacje gęstości *udanych wyjść z poziomu* dla każdego algorytmu.

ścią na parametry. Najszerszy i przesunięty ku niższym wartościom rozkład występuje w *Automatach Komórkowych*, wskazując na największą zmienność i częstsze niepowodzenia.

4.8.5 Stosunek liniowych korytarzy — charakterystyka

Wskaźnik „stosunek liniowych korytarzy” opisuje udział pól przechodnich należących do odcinków prostych w stosunku do całej powierzchni przechodniej. Za odcinki proste uznaje się ciągi komórek korytarza, w których pole ma dwóch sąsiadów ułożonych współliniowo (w poziomie lub pionie).

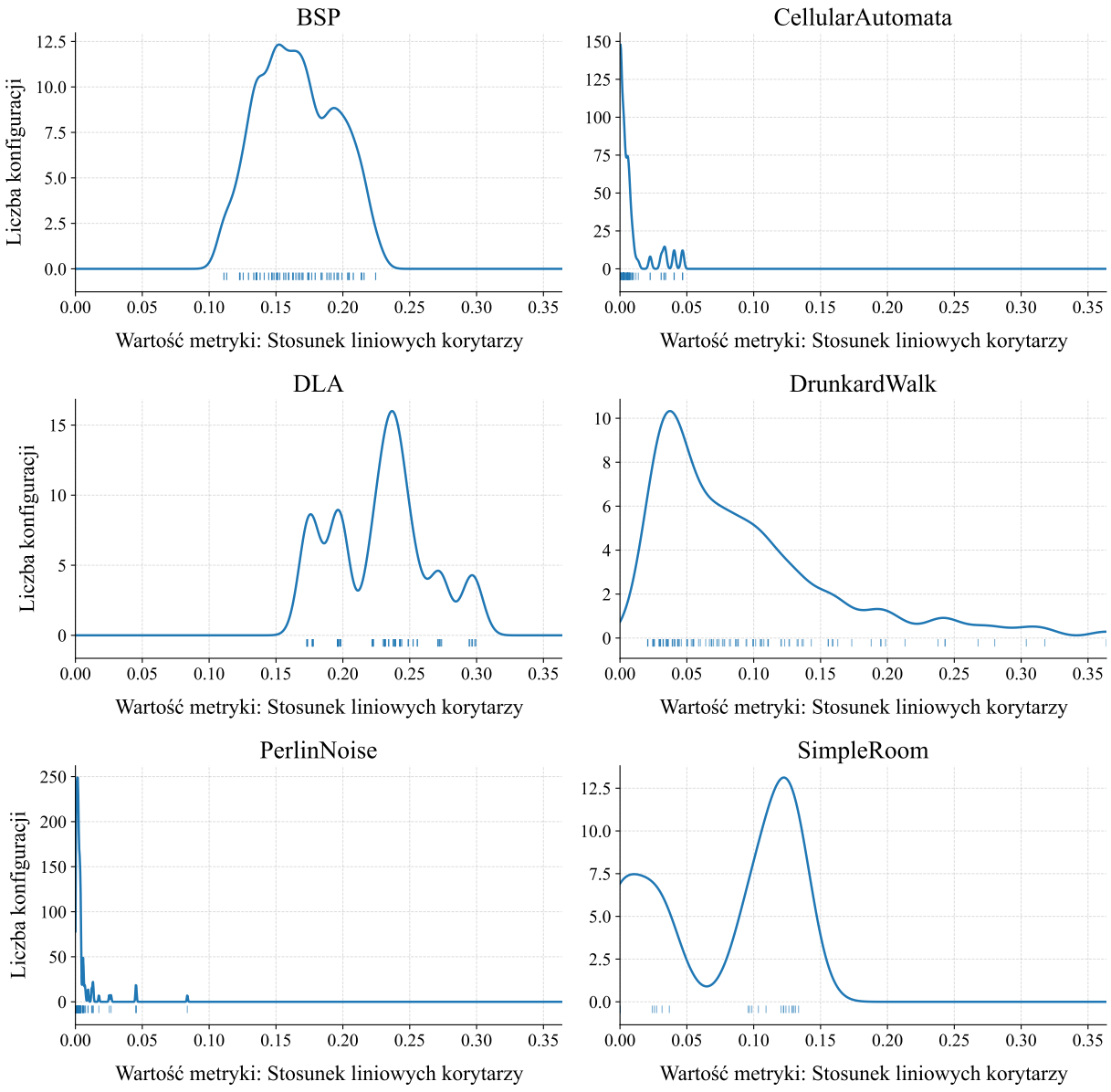


Rysunek 4.36: Średnie wyniki stosunku liniowych korytarzy dla badanych algorytmów. Wąsy przedstawiają 95% przedziały ufności.

Wyniki zbiorcze i ich interpretacja.

Porządek średnich (rys. 4.36) wskazuje na wyraźne różnice między podejściami. Najwyższe wartości uzyskuje *DLA*, w którym często tworzą się długie odcinki korytarzy. Nieco niższy, ale nadal wysoki udział odcinków prostych obserwuje się w *BSP*; regularny podział przestrzeni sprzyja powstawaniu korytarzy o prostoliniowym przebiegu między salami tak samo jak wymuszone połaczenia pomiędzy nimi. *Pijany Spacer* lokuje się pośrodku zestawienia — losowy marsz generuje mieszankę odcinków prostych i łagodnych załamań, co daje umiarkowany udział korytarzy liniowych. W algorytmie *Prosty Pokój* wartość wskaźnika jest niewielka, ponieważ struktura poziomu dominuje pojedynczą salą, a łączniki (jeśli występują) są krótkie. Najniższe udziały notują *Automaty Komórkowe* i *Szum Perlina*: jaskiniowe, organiczne geometrie tych algorytmów przynoszą korytarze przestrzenie, o licznych przewężeniach i rozgałęzieniach, w których odcinki idealnie proste stanowią jedynie śladową część.

Rozkłady gęstości (rys. 4.37) potwierdzają opis powyżej. W *BSP* krzywa jest stosunkowo wąska i jednomodalna — udział odcinków prostych pozostaje stabilny w szerokim zakresie konfiguracji. *DLA* ujawnia kilka wyraźnych wierzchołków, co wskazuje na współistnienie odmiennych reżimów wzrostu: od wariantów z licznymi prostymi „ramionami” po bardziej rozgałęzione układy. *Pijany Spacer* ma rozkład szeroki i prawostronnie wydłużony: większość konfiguracji daje niewielkie lub umiarkowane wartości, lecz pojawiają się także przypadki z wyraźnie dłuższymi odcinkami prostymi. W przypadku algorytmu *Prosty Pokój* rozkład jest dwuwierzchołkowy — obok konfiguracji bez korytarzy prostych



Rysunek 4.37: Jądrowe estymacje gęstości *stosunku liniowych korytarzy* dla każdego algorytmu.

pojawiają się sytuacje, w których łączniki między punktami specjalnymi tworzą krótkie, regularne odcinki. *Automaty Komórkowe* oraz *Szum Perlina* skupiają się silnie przy wartościach bliskich zera.

Implikacje dla projektowania i dalszych porównań. Wysoki stosunek liniowych korytarzy sprzyja przejrzystości i przewidywalności ruchu, ale może też wydłużać trasy przez sekwencje długich, prostych odcinków. Z kolei wartości niskie oznaczają układy bardziej kręte i rozgałęzione, zwiększające lokalną złożoność nawigacji. W ujęciu porównawczym wskaźnik ten dobrze uzupełnia metryki „ścieżki krytycznej" oraz „kroków do wyjścia": podobny poziom trudności może wynikać z długości prostych sekwencji.

4.9 Współzależności pomiędzy metrykami

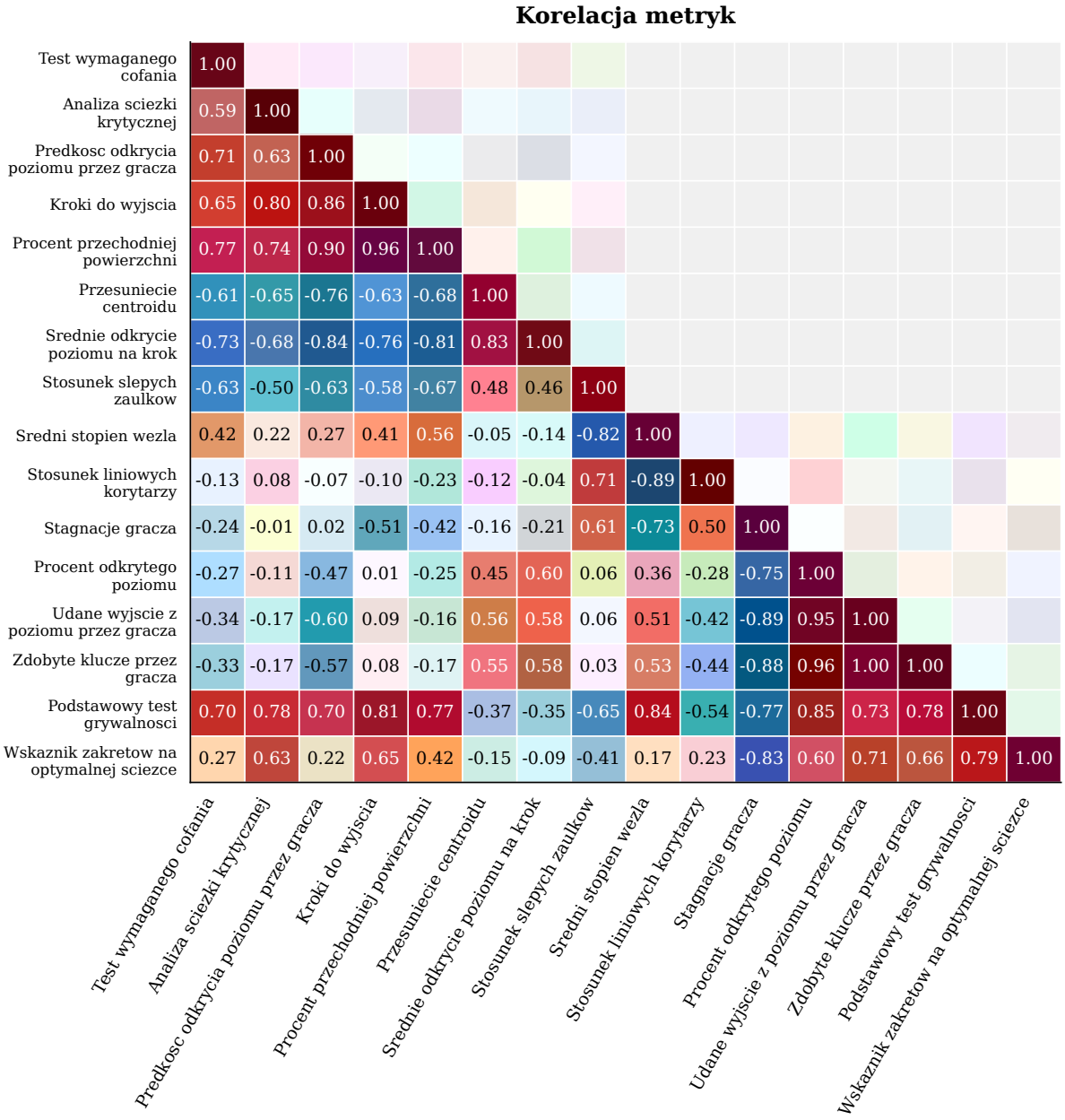
W tej sekcji zbadane zostały współzależności pomiędzy metrykami opisującymi strukturę poziomów oraz zachowanie agenta. Dzięki temu możemy rozróżnić które z nich opisują ten sam konstrukt, które się uzupełniają, a które mogą prowadzić do podobnych wniosków mimo różnej nazwy. Dla każdej pary metryk oszacowano korelację *Pearsona*, a następnie zebrano wyniki z wielu konfiguracji i algorytmów celem wizualizacji (rys. 4.38).

Zależności między metrykami

Wskaźnik ukończenia poziomu, *wskaźnik zdobycia klucza* oraz *odkryty obszar* są na ogół silnie zależne od siebie. Gdy rośnie skuteczność w znajdowaniu klucza i docieraniu do wyjścia, zwykle rośnie też odsetek odkrytych pól. W praktyce opisują wspólny aspekt — czy agent faktycznie posuwa się naprzód. Uzupełniająco warto śledzić *prędkość odkrycia poziomu przez gracza* oraz *średnie odkrycie poziomu na krok*, które wprowadzają wymiar czasowy i wydajnościowy eksploracji. Poziomy o wysokim *procencie przechodniej powierzchni* sprzyjają szybkim przyrostom odkrycia na starcie, ale zarazem wydłużają obowiązkową trasę $S \to K \to E$. Oznacza to, że w układach bardziej otwartych agent łatwiej trafia na nowe pola, lecz dystans między punktami krytycznymi bywa większy. Wzrost *stosunku liniowych korytarzy* i *stosunku ślepych zaułków* koreluje z większym ryzykiem *stagnacji gracza* i niższym udziałem udanych przejść, ponieważ agent częściej sprawdza ślepe gałęzie i trudniej mu wracać na efektywne trajektorie. Działa tu przeciwwaga w postaci *średniego stopnia węzła* - większa liczba alternatywnych przejść zmniejsza podatność na zacięcia.

Wnioski

Same korelacje nie rozstrzygają o przyczynowości, lecz pozwalają lepiej uchwycić, które aspekty poziomów i zachowania są ze sobą powiązane. W praktyce, przy projektowaniu



Rysunek 4.38: Mapa korelacji Pearsona pomiędzy metrykami geometrii i zachowania agenta (wszystkie algorytmy i konfiguracje).

eksperymentów i interpretacji wyników, warto dobierać metryki wzajemnie się uzupełniające. W obszarze postępu wystarczy jeden reprezentatywny wskaźnik (np. odsetek ukończonych poziomów) aby oddać główny sens zmian. Długość obowiązkowej drogi można zwięźle opisywać metrykami ścieżki krytycznej, zamiast szczegółowo zliczać kroki do wyjścia. Dynamikę eksploracji warto przedstawiać dwutorowo: jako tempo w czasie oraz jako efektywność na krok, ponieważ te dwa wymiary bywają rozbieżne. Dla struktury lokalnej użyteczne są miary udziału ślepych zaułków i odcinków prostych lecz można je zastąpić ujęciem bardziej topologicznym (np. średnim stopniem węzła). Obraz dopełnia informacja o rozległości przestrzeni przechodniej, która stanowi ważną zmienną moderującą wiele zależności.

4.10 Podsumowanie badań

Przeprowadzone badania umożliwiły kompleksowe porównanie wybranych algorytmów pod kątem ich skuteczności, wydajności oraz praktycznego zastosowania. Analiza wyników pokazała, że każdy z rozważanych algorytmów posiada zarówno swoje mocne strony, jak i ograniczenia, co w istotny sposób wpływa na ich użyteczność w różnych scenariuszach. Wskazuje to, że wybór odpowiedniego rozwiązania powinien być uzależniony od specyficznych wymagań środowiska, w którym algorytm ma być stosowany. Istotnym rezultatem badań było także wykazanie różnic w zachowaniu algorytmów przy zmianie parametrów wejściowych. Niektóre metody wykazały się większą stabilnością i odpornością na zmiany danych, podczas gdy inne charakteryzowały się znaczną wrażliwością, co może utrudniać ich zastosowanie w warunkach rzeczywistych. Porównanie to pozwoliło wskazać algorytmy bardziej uniwersalne, a także takie, które sprawdzają się jedynie w ściśle określonych przypadkach.

W przeprowadzonych eksperymentach stwierdzono, że najwyższy odsetek udanych ukończeń poziomu uzyskiwały układy generowane metodami *BSP* oraz *Pijany Spacer*, co wskazuje na ich wysoką niezawodność przy niskich wrażliwościach dotyczących strojenia parametrów. Najkrótszą ścieżkę krytyczną zapewniał *Prosty Pokój*, podczas gdy układy powstałe w wyniku działania algorytmów *BSP* oraz *Pijany Spacer* były nieco dłuższe, lecz nadal względnie zwarte. Najdłuższe przejścia, mierzone liczbą „kroków do wyjścia", obserwowano w przypadku *Szumu Perlina* oraz *DLA*, co wiąże się z większą rozległością i rozgałęzieniem topologii tych poziomów. Analiza liniowości korytarzy wykazała ponadto, że *DLA* i *BSP* generują najwyższy udział długich odcinków prostych — sprzyjających czytelności nawigacji, lecz mogących wydłużać trasy — natomiast *Automaty Komórkowe* i *Szum Perlina* prowadzą do układów bardziej krętych i rozgałęzionych (o niższej liniowości), typowych dla form jaskiniowych i cechujących się większą złożonością nawigacyjną.

Podsumowując, uzyskane wyniki jednoznacznie pokazują, że nie istnieje jedno rozwiązanie dominujące we wszystkich aspektach. Każdy z badanych algorytmów znajduje

swoje optymalne zastosowanie w zależności od przyjętych kryteriów oceny, charakterystyki danych oraz wymagań praktycznych. Wyniki badań mogą stanowić punkt wyjścia do dalszych analiz porównawczych, a także podstawę do wyboru najodpowiedniejszej metody PCG przez projektantów poziomów.

Rozdział 5

Podsumowanie

Praca porównuje wybrane algorytmy proceduralnego generowania poziomów w grach roguelike, oceniane przy użyciu ujednoliconego zestawu metryk strukturalnych i grywalnościowych, w tym miar wyznaczanych na podstawie symulacji gracza. Autor dobrał reprezentatywne metody, zaprojektował wspólny zestawu metryk i kryteriów, przygotował zautomatyzowane środowisko badawcze z symulacją wirtualnego gracza, zrealizował szeroką seriię eksperymentów oraz opracował wyniki.

Materiał badawczy objął algorytmy: *Pijany Spacer*, *Binarny Podział Przestrzeni*, *Automaty Komórkowe*, *Szum Perlina* oraz *Agregację Ograniczoną Dyfuzją*, a także algorytm bazowy *Prosty Pokój*. Każdy algorytm analizowano w szerokim zakresie parametrów, aby uchwycić zróżnicowanie zachowań i zależności między konfiguracjami a wartościami metryk w zintegrowanym środowisku, które automatycznie generowało poziomy, obliczało metryki, uruchamia symulacje i agregację wyników. Powtarzalność zapewniono dzięki kontroli źródeł losowości i jednolitym procedurom raportowania; te same dane wejściowe prowadzą do tych samych tabel i rysunków.

Cel pracy dyplomowej został osiągnięty. Dzięki zastosowaniu powtarzalnych procedur, spójnych metryk oraz jednolitego środowiska, opracowano porównawczą i ilościową analizę obraz właściwości algorytmów PCG stosowanych w grach *roguelike*. Analiza ujawniła profilowe mocne i słabe strony poszczególnych metod oraz pozwoliła sformułować praktyczne wskazówki dla twórców i projektantów poziomów. Wyniki nie potwierdzają istnienia algorytmu dominującego we wszystkich kryteriach, każda metoda ma specyficzne zalety i ograniczenia, które należy rozpatrywać w świetle wymagań projektu i zamierzonych doświadczeń gracza. Wybór algorytmu powinien być więc świadomy i kontekstowy, a nie oparty na ogólnych przesłankach o jego wyższości. Przedstawione rezultaty stanowią solidny fundament dla dalszych, systematycznych badań i wdrożeń oraz mogą służyć jako praktyczny przewodnik przy doborze metod PCG do konkretnych celów i zastosowań.

Ograniczenia i kierunki dalszych badań

Wnioski empiryczne odnoszą się do zachowania jednego typu agenta oraz do ustalonego wymiaru siatki, co ogranicza zakres uogólnień. Wartości metryk należy traktować jako miary względne, przydatne przede wszystkim do porównań i analiz wrażliwości w obrębie zadanej konfiguracji, a nie jako bezwzględny opis jakości całej rozgrywki.

W pracy przyjęto stały rozmiar planszy, co uzasadniono kompromisem między rozdzielczością a kosztami obliczeń, lecz takie założenie może wprowadzać efekt skali oraz wpływać na stabilność wyników w pobliżu krawędzi planszy. Z tego względu sensowna kontynuacja badań wymagałaby jednoczesnego zagęszczenia próbkowania przestrzeni parametrów oraz systematycznego sprawdzenia, czy obserwowane zależności utrzymują się dla różnych rozmiarów siatki. Gęstsze i szersze zakresy wartości pozwoliłyby uchwycić lepiej nieliniowe powiązania oraz interakcje między poszczególnymi algorytmami, natomiast testy na wielu wymiarach początkowych poziomu umożliwiłyby ocenę odporności wniosków na zmianę skali i stopień wrażliwości na zjawiska brzegowe.

Dodatkowym ograniczeniem jest oparcie ewaluacji na jednej polityce działania wirtualnego gracza, podczas gdy rzeczywiste zachowania mogą odpowiadać wielu profilom decyzyjnym. Włączenie do analiz wariantów bardziej zachłannych, ostrożnych, ryzykujących, losowych oraz uczących się pozwoliłoby ocenić, czy wyniki są specyficzne dla przyjętej strategii, czy też zachowują ważność w szerszym spektrum uwarunkowań behawioralnych graczy. Dopiero połączenie gęstego próbkowania parametrów, przeglądu różnych rozmiarów siatki oraz porównania wielu polityk działania gracza dałoby podstawę do mocniejszych uogólnień i pełniejszej charakterystyki badanych algorytmów.

Bibliografia

[1] Zeyad Abd Algfoor, Mohd Shahrizal Sunar i Hoshang Kolivand. „A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games". W: *International Journal of Computer Games Technology* 2015.1 (2015), s. 736138. DOI: https://doi.org/10.1155/2015/736138. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2015/736138. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2015/736138.

[2] Wirawan Istiono Andy Koesnaedi. „Implementation Drunkard's Walk Algorithm to Generate Random Level in Roguelike Games." W: (2022), Volume 5, Issue 2, pp. 97–103. ISSN: 2581-6187.

[3] Jessica Baron. „Procedural Dungeon Generation Analysis and Adaptation". W: kwiecień 2017, s. 168–171. DOI: 10.1145/3077286.3077566.

[4] *Cellular Automata Method for Generating Random Cave-Like Levels*. 2023. URL: https://www.roguebasin.com/index.php/Cellular_Automata_Method_for_Generating_Random_Cave-Like_Levels.

[5] Yu-Cheng Cheng, Yanan Wang i Yan Pei. „User Experience Evaluation in Game Content Generation of Super Mario Bros." W: *Innovative Computing 2025, Volume 4*. Oprac. Hao-Shang Ma, Hwa-Young Jeong, Yu-Wei Chan i Hsuan-Che Yang. Singapore: Springer Nature Singapore, 2025, s. 42–48. ISBN: 978-981-96-8011-5.

[6] Laurence Murray Johnson, Georgios N. Yannakakis i Julian Togelius. „Cellular automata for real-time generation of infinite cave levels". W: *PCGames*. 2010. URL: https://api.semanticscholar.org/CorpusID:207179823.

[7] N.G. VAN KAMPEN. *Stochastic Processes in Physics and Chemistry*. Elsevier, 2007. ISBN: 978-0-444-52965-7. DOI: https://doi.org/10.1016/B978-0-444-52965-7.X5000-4.

[8] Marek Kopel i Grzegorz Maciejewski. „Comparison of Procedural Noise-Based Environment Generation Methods". W: *Computational Collective Intelligence*. Cham: Springer International Publishing, 2020, s. 878–887. ISBN: 978-3-030-63007-2.

[9] Antonios Liapis, Georgios Yannakakis i Julian Togelius. „Towards a Generic Method of Evaluating Game Levels". W: *Proceedings of the 9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2013* 9 (styczeń 2013), s. 30–36. DOI: 10.1609/aiide.v9i1.12680.

[10] Jacques Brancher Luiz Rodrigues Robson Bonidia. „Procedural versus human level generation: Two sides of the same coin?" W: *International Journal of Human-Computer Studies* (2020). Accessed: 2025-08-21. DOI: https://doi.org/10.1016/j.ijhcs.2020.102465. URL: https://www.sciencedirect.com/science/article/pii/S1071581920300677.

[11] Christopher Cleghorn Michael Beukman Steven James. „Towards Objective Metrics for Procedurally Generated Video Game Levels". W: Accessed: 2025-08-04. 2022. DOI: https://doi.org/10.48550/arXiv.2201.10334. URL: https://arxiv.org/abs/2201.10334.

[12] NIRA RICHTER-DYN NARENDRA S. GOEL. *Stochastic Models in Biology*. Elsevier, 1974. ISBN: 978-0-12-287460-4. DOI: https://doi.org/10.1016/C2013-0-10736-2.

[13] Julian Togelius Noor Shaker i Mark J. Nelson. *Procedural Content Generation in Games*. Switzerland: Springer, 2016. ISBN: 978-3-319-42714-0. DOI: 10.1007/978-3-319-42716-4.

[14] Raffaella Nori, Micaela Maria Zucchelli, Massimiliano Palmiero i Laura Piccardi. „Environmental cognitive load and spatial anxiety: What matters in navigation?" W: *Journal of Environmental Psychology* 88 (2023), s. 102032. ISSN: 0272-4944. DOI: https://doi.org/10.1016/j.jenvp.2023.102032. URL: https://www.sciencedirect.com/science/article/pii/S0272494423000804.

[15] Laurissa Tokarchuk Oliver Withington Michael Cook. „On the Evaluation of Procedural Level Generation Systems". W: *FDG '24: Proceedings of the 19th International Conference on the Foundations of Digital Games* 22 (2024), s. 1–10. DOI: 10.1145/3649921.3650016.

[16] Szabolcs Orbán i Szabados. „Roguelike games: The way we play". W: *International Journal of Engineering and Management Sciences* (2022), s. 80–92. DOI: 10.21791/IJEMS.2022.4.7.

[17] *Procedural Dungeon Generation: An In-Depth Guide to Random Walk Algorithms - Linux Code*. 2024. URL: https://thelinuxcode.com/procedural-dungeon-generation-an-in-depth-guide-to-random-walk-algorithms/.

[18] Pratama Aditya Putra, Jos Timanta Tarigan i Elviawaty Muisa Zamzami. „Procedural 2D Dungeon Generation Using Binary Space Partition Algorithm And L-Systems". W: *2023 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*. 2023, s. 365–369. DOI: 10.1109/IC3INA60834.2023.10285811.

[19] Hannes Risken. *The Fokker-Planck Equation*. Berlin: Springer Berlin, Heidelberg, 1996. ISBN: 978-3-540-61530-9. DOI: https://doi.org/10.1007/978-3-642-61544-3.

[20] *Roguebasin - Berlin Interpretation*. 2024. URL: https://www.roguebasin.com/index.php/Berlin_Interpretation.

[21] Bartłomiej Szabat i Małgorzata Plechawska-Wójcik. „Comparative Analysis of Selected Game Engines". W: *Journal of Computer Sciences Institute* 29 (grudzień 2023), s. 312–316. DOI: 10.35784/jcsi.3771.

[22] William C. Thibault i Bruce F. Naylor. „Set operations on polyhedra using binary space partitioning trees". W: *SIGGRAPH Comput. Graph.* 21.4 (sierpień 1987), 153–162. ISSN: 0097-8930. DOI: 10.1145/37402.37421. URL: https://doi.org/10.1145/37402.37421.

[23] T. A. Witten i L. M. Sander. „Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon". W: *Physical Review Letters* (1983). DOI: https://doi.org/10.1103/PhysRevB.27.5686.

[24] B. Yamauchi. „A frontier-based approach for autonomous exploration". W: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*. 1997, s. 146–151. DOI: 10.1109/CIRA.1997.613851.

Dodatki

Spis skrótów i symboli

PCG	proceduralne generowanie treści (ang. *procedural content generation*)
BSP	binarny podział przestrzeni (ang. *binary space partitioning*)
DLA	agregacja ograniczona dyfuzją (ang. *diffusion-limited aggregation*)
CA	automat komórkowy (ang. *cellular automata*)
A*	algorytm wyznaczania ścieżki (ang. *A-star*)
FOV	pole widzenia (ang. *field of view*)
NPC	bohater niezależny (ang. *non-player character*)
CI	przedział ufności (ang. *confidence interval*)
SEM	błąd standardowy średniej (ang. *standard error of the mean*)
KDE	jądrowa estymacja gęstości (ang. *kernel density estimation*)
JSON	format danych (ang. *JavaScript Object Notation*)
AUC	pole pod krzywą (ang. *Area Under the Curve*)
S	węzeł startowy (punkt startu gracza)
K	węzeł klucza (położenie klucza)
E	węzeł wyjścia (położenie wyjścia z poziomu)

Lista dodatkowych plików, uzupełniających tekst pracy

Do pracy dołączono dodatkowe pliki obejmujące dane eksperymentalne, wyniki analiz oraz środowisko badawcze, umożliwiające powtórzenie przeprowadzonych badań:

- **Dane z Eksperymentów** – surowe dane uzyskane podczas przeprowadzonych eksperymentów.
- **Dane z Analizy** – opracowane wyniki analiz, obejmujące:
  - korelacje metryk i parametrów,
  - analizy par parametrów i metryk,
  - analizy rozkładów,
  - zestawienia uśrednionych wartości.
- **Środowisko Badawcze** – kompletne środowisko wykorzystywane w pracy:
  - *Warstwa Analityczno-Wizualna* – skrypty w języku *Python* do analizy i wizualizacji wyników,
  - *Warstwa Eksperymentalna* – projekt w silniku *Unity* zawierający skrypty, pakiety oraz ustawienia projektu.

Spis rysunków