Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Stanisław Frejlak
Nr albumu: 371283

Yose KataGo, czyli teoria końcowej fazy gry Go a programy wykorzystujące głębokie sieci rezydualne

Praca licencjacka
na kierunku MATEMATYKA

Praca wykonana pod kierunkiem
dr. Marcina Szczuki
Instytut Informatyki

Listopad 2020

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

Streszczenie

Go jest skomplikowaną grą logiczną, która przez lata stanowiła wyzwanie dla sztucznej inteligencji. Końcowa faza gry Go poddaje się opisowi formalnemu. Była ona jedną z inspiracji dla twórców kombinatorycznej teorii gier. Teoria ta pozwala na bardziej precyzyjną analizę sytuacji niż klasyczne metody stosowane przez graczy Go. Nie wystarcza ona jednak do stworzenia silnych programów grających. Opanowanie gry na poziomie mistrzowskim przez programy stało się możliwe dopiero dzięki zastosowaniu sieci neuronowych oraz algorytmu MCTS, bez implementowania wypracowanej teorii. W pracy badam, czy programy takie dobrze rozumieją teorię fazy końcowej.

Słowa kluczowe

Go, kombinatoryczna toeria gier, sieci konwolucyjne, Monte Carlo Tree Search

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.1 Matematyka

Klasyfikacja tematyczna

68. Computer science
68.T. Artificial intelligence
68.T.05. Learning and adaptive systems in artificial intelligence

Spis treści

Wprowadzenie

Azjatycka gra Go jest jedną z najstarszych, a zarazem najtrudniejszych gier logicznych. Gra polega na stawianiu na planszy kamieni w celu otoczenia większego terytorium niż przeciwnik.

Przez wiele lat stworzenie silnego programu grającego w Go stanowiło duże wyzwanie dla programistów i badaczy zajmujących się sztuczną inteligencją. W trakcie partii Go gracz za każdym razem, mając wykonać ruch, staje przed wyborem jednej spośród około dwustu możliwości. Rozgrywka jest przez to bardzo skomplikowana. Klasyczne algorytmy przeszukiwania drzewa gry nie są w stanie znaleźć dobrych zagrań nawet przy użyciu dużej mocy obliczeniowej. Gracze Go przy podejmowaniu decyzji w dużej mierze kierują się intuicją. Programiści zajmujący się komputerowym Go nie znaleźli sposobu na zalgorytmizowanie tych intuicji.

Niektóre aspekty Go poddają się opisowi matematycznemu. Dla ruchów w końcowej fazie gry istnieje nieskomplikowany arytmetyczny wzór, który wycenia ich wartość [17]. Doświadczeni zawodnicy potrafią się nim sprawnie posługiwać i wykorzystują obliczone wartości przy podejmowaniu decyzji. Dzięki temu faza końcowa rodzi największe nadzieje na stworzenie algorytmu wybierania optymalnych ruchów.

W drugiej połowie XX wieku prace Johna Conwaya, Elwyna Berlekampa i Richarda Guya [7, 3] zapoczątkowały nową dziedzinę matematyki, kombinatoryczną teorię gier. Jedną z inspiracji dla nowej teorii były właśnie końcówki w grze Go. Kombinatoryczna teoria gier dostarczyła silnych narzędzi, które pozwoliły na dokładniejszą analizę pozycji niż klasyczny sposób liczenia wartości ruchów.

Spektakularną demonstracją siły wypracowanych narzędzi matematycznych było stworzenie przez Elwyna Berlekampa i Davida Wolfe'a *problemu peszącego zawodowców* (ang. *9-dan stumping problem*) [4]. Jest to złożona pozycja z końcowej fazy gry, w której nawet najsilniejsi zawodnicy nie potrafią znaleźć optymalnego wariantu dla obu stron. Tymczasem zastosowanie kombinatorycznej teorii gier pozwala na względnie szybką precyzyjną analizę sytuacji. Badacze teorii żywili nadzieję, że jej dalszy rozwój doprowadzi do przełomu w komputerowym Go [11, 4].

Przełom w komputerowym Go nastąpił w 2016 roku, kiedy program AlphaGo pokonał czołowego gracza na świecie, Lee Sedola. Zespół DeepMind do stworzenia AlphaGo nie wykorzystał jednak osiągnięć teorii gier. Kluczem okazało się zastosowanie sieci neuronowych w połączeniu z algorytmem Monte Carlo Tree Search. Do trenowania sieci użyto uczenia pod nadzorem oraz ze wzmocnieniem. Stworzony rok później program AlphaGo Zero [14] osiągnął poponadludzki poziom, ucząc się od zera: bez wykorzystania wiedzy eksperckiej ani przykładowych partii silnych graczy. Sukces zespołu DeepMind został uznany za wielki krok naprzód w badaniach nad sztuczną inteligencją.

Algorytm użyty w programie AlphaGo Zero doczekał się kilku otwartoźródłowych implementacji. David Wu, twórca programu KataGo, zmodyfikował algorytm, aby wykorzystać różne elementy mechaniki gry. Użyta metoda pozwoliła na wielokrotne przyspieszenie treningu sieci [19]. Jedna z dodanych funkcjonalności polegała na przewidywaniu terytoriów

graczy. Dzięki temu nie tylko KataGo szybciej uczyło się dobrych zagrań, ale również użytkownicy programu więcej dowiedzieli się o tym, jak sieć neuronowa ocenia pozycje na planszy.

Przewidywanie terytoriów przez KataGo pozwala przetestować, jak dobre rozumienie teorii końcówek osiągnął program. Niniejsza praca przedstawia wyniki takich testów. Sprawdziłem, czy KataGo potrafi precyzyjnie ocenić wartości ruchów w końcówce. Następnie zbadałem, czy program w końcówce wybiera zagrania zgodne z algorytmem bazującym na kombinatorycznej teorii gier. W ramach ostatniego testu przyjrzałem się, czy problem peszący zawodowców zapeszy również KataGo.

W rozdziale 1 w zwięzły sposób wyjaśniam zasady Go. W rozdziale 2 omawiam klasyczną metodę liczenia wartości ruchów oraz zastosowania kombinatorycznej teorii gier w końcowej fazie gry. W rozdziale 3 pokrótce przedstawiam parę rozwiązań programistycznych, które doprowadziły do powstania programów ogrywających najsilniejszych goistów na świecie. W rozdziale 4 opisuję testy, przy pomocy których sprawdziłem, jak dobrze KataGo rozumie teorię końcówek. W rozdziale 5 dokonuję interpretacji uzyskanych wyników i podsumowuję pracę.

W pracy używam kilku pojęć z żargonu goistycznego. Przy pierwszym użyciu pojęcia oznaczam je kursywą oraz krótko tłumaczę jego znaczenie. Tłumaczenia pojęć są również dostępne w słowniczku na końcu pracy.

Rozdział 1

Gra Go

Go jest grą planszową dla dwóch graczy. Pochodzi ona z Chin i ma historię liczącą kilka tysięcy lat. Jest to gra o doskonałej informacji bez czynnika losowego. Zasady Go są bardzo proste. Jednak opanawonie gry na poziomie mistrzowskim stanowi zadanie na wiele lat życia. Go odznacza się dużym stopniem skomplikowania. Liczba możliwych rozgrywek jest w przybliżeniu równa 10^{360}, podczas gdy np. w szachach ta liczba to 10^{123} [1].

Istnieje kilka zestawów reguł Go, które minimalnie się od siebie różnią. Dla celów tej pracy przedstawiam reguły japońskie.

W podrozdziałach 1.1 i 1.2 przedstawiam podstawowe zasady gry. Tłumaczę, na czym polega ruch, jaki jest cel gry oraz czym jest zbijanie kamieni. W podrozdziale 1.4 tłumaczę procedurę liczenia wyniku. W podrozdziale 1.5 opisuję, jak zasady Go radzą sobie z możliwością zapętlenia się gry. W podrozdziałach 1.3 i 1.6 prezentuję podstawową mechanikę gry wynikającą ze sformułowania reguł.

1.1. Przebieg rozgrywki

Plansza do Go jest kwadratową siatką pionowych i poziomych linii. Pola planszy to punkty przecięcia tych linii. Partie turniejowe rozgrywa się na planszach 19×19. Czasem używane są również plansze o wymiarach 13×13 i 9×9, zwłaszcza przez graczy początkujących.

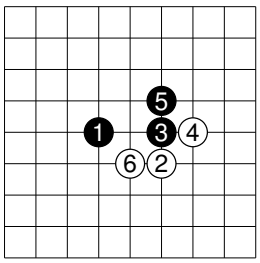


Diagram 1.1: Przykładowe pierwsze ruchy gry

Partię Go rozgrywa dwóch graczy. Jeden dysponuje zestawem czarnych, a drugi zestawem białych kamieni. Graczy często skrótowo nazywa się *Czarne* i *Białe*. Na początku gry plansza jest pusta. Pierwszy ruch należy do Czarnych. Gracze wykonują ruchy na przemian. Ruch polega na położeniu jednego ze swoich kamieni na jednym z niezajętych pól bądź spasowaniu (p. diagram 1.1). Jeśli jeden z graczy spasuje i w następnym ruchu spasuje jego przeciwnik, gra dobiega końca. Zwycięzcą zostaje gracz, który uzyskał kontrolę nad większą częścią planszy niż przeciwnik (dokładniejsze tłumaczenie w podrozdziale 1.4).

1.2. Zbijanie kamieni

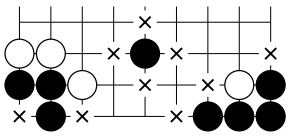
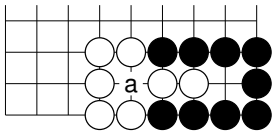


Diagram 1.2: Łańcuchy czarnych kamieni mają odpowiednio: 2, 4 i 3 oddechy.

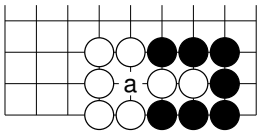
Pola sąsiadujące z danym polem to te, które są z nim połączone linią. Ściślej, są to pola, których współrzędne różnią się od współrzędnych danego pola na dokładnie jednej pozycji o dokładnie 1. Kamienie tego samego koloru położone na sąsiadujących polach nazywa się *połączonymi*. *Łańcuch* to zbiór połączonych kamieni. Formalnie, bycie połączonym jest relacją równoważności, a łańcuch to klasa abstrakcji tej relacji. Niezajęte pola sąsiadujące z kamieniami należącymi do łańcucha nazywa się *oddechami* tego łańcucha (p. diagram 1.2).

Kamień raz postawiony na planszy nigdy nie jest przesuwany. Może jednak zostać zdjęty z planszy. Następuje to, kiedy łańcuch, do którego należy dany kamień, traci swój ostatni oddech. Jeśli gracz stawia kamień na polu, które było jedynym oddechem łańcucha przeciwnika, to mówi się, że zbija ten łańcuch. Zdejmuje wtedy z planszy zbite kamienie. Ruch, który grozi zbiciem łańcucha, to znaczy zabiera mu przedostatni oddech, nazywa się *atari*.

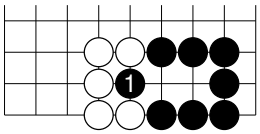
Jeśli po ruchu gracza jego własny łańcuch nie miałby żadnego oddechu, to taki ruch nazywa się *samobójczym*. Ruchy samobójcze są niedzwolone. Na diagramach 1.3a i 1.3b pole ***a*** jest otoczone przez Białe. Ruch Czarnych na diagramie 1.3a byłby samobójczy. Jednak na diagramie 1.3b łańcuch dwóch białych kamieni ma tylko jeden oddech (mówi się, że *stoi pod atari*). Kiedy Czarne grają w ***a***, zbijają ten łańcuch. Jak pokazuje diagram 1.3c, po zdjęciu białych kamieni z planszy kamień ❶ posiada jeden oddech. Dlatego ruch Czarnych w tym przypadku nie jest samobójczy.



(a) Ruch Czarnych w ***a*** jest samobójczy.



(b) Ruch Czarnych w ***a*** jest dozwolony.



(c) Sytuacja po ruchu Czarnych w ***a***.

Diagram 1.3: Zasada zbijania

1.3. Życie i śmierć - konsekwencja zasady zbijania

W Go istnieją łańcuchy, których przeciwnik nigdy nie będzie w stanie zbić. Na diagramie 1.4 oba łańcuchy Czarnych są otoczone od zewnątrz przez Białe. Łańcuch po lewej otacza jedno wolne przecięcie *a* (takie przecięcie nazywa się *okiem* łańcucha). Ponieważ jest to jedyny oddech łańcucha, Białe mogą tam zagrać i zbić Czarne. Łańcuch Czarnych na środku posiada dwoje oczu. Każdy z ruchów *b* i *c* jest samobójczy dla Białych. Dlatego Białe nigdy nie odbiorą oddechów czarnemu łańcuchowi. Mówi się, że taki łańcuch jest *żywy*. Posiadanie dwóch oczu zapewnia życie.

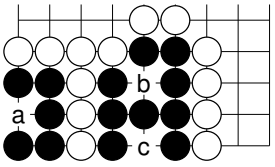


Diagram 1.4: Ruch Białych *a* zbija łańcuch po lewej. Ruchy *b* i *c* są samobójcze.

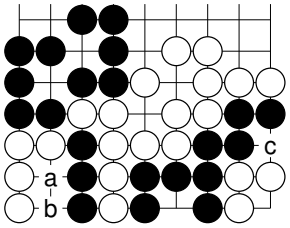


Diagram 1.5: Życie bez dwojga oczu

Jedno oko nie zapewnia życia, jak pokazał przykład łańcucha na diagramie 1.4. Kamienie, które niezależnie od starań gracza będą mogły zostać zbite przez przeciwnika, nazywa się *martwymi*. Kamienie już zbite również są martwe. Na diagramie 1.4 ruch Czarnych w *a* byłby samobójczy. Tymczasem Białe mogą tam zagrać w dowolnym momencie i zbić Czarne. Dlatego czarny łańcuch w narożniku jest martwy.

Diagram 1.5 pokazuje inne przykłady żywych kamieni. Dwa łańcuchy Białych i dwa łańcuchy Czarnych w dolnych narożnikach nie posiadają dwojga oczu. Niemniej, przeciwnik nie może doprowadzić do ich zbicia. Jeśli któryś z graczy zabrałby oddech jednemu z tych łańcuchów, postawiłby własne kamienie w atari. Wówczas przeciwnik zbiłby te kamienie, zapewniając życie własnemu łańcuchowi.

1.4. Liczenie wyniku

Podczas partii Go celem gracza jest zdobycie kontroli nad większą częścią planszy niż przeciwnik. Kiedy gracze widzą, że plansza została już podzielona na terytoria Białych i Czarnych, pasują. Terytorium gracza jest to zbiór przecięć otoczonych przez jego żywe kamienie. Formalnie, przecięcie należy do terytorium gracza, jeśli jest albo puste, albo zajęte przez martwy kamień przeciwnika, a każde przecięcie z nim sąsiadujące albo też należy do tego terytorium, albo jest zajęte przez żywy kamień gracza. Po tym jak obaj gracze spasują jeden po drugim, przystępują do policzenia wyniku gry.

Gracz otrzymuje punkt za każde wolne przecięcie należące do jego terytorium oraz za każdy martwy kamień przeciwnika[1]. Wynik gry jest równy różnicy liczby punktów obu graczy.

W Go Czarne posiadają przewagę ze względu na wykonywanie pierwszego ruchu. Dlatego gry turniejowe od XX wieku rozgrywa się z zasadą *komi*. Komi jest to pewna ustalona liczba dodatkowych punktów, które otrzymują Białe. W zasadach japońskich komi jest równe $6,5$ punktu. Połówka punktu sprawia, że partia nigdy nie kończy się remisem.

[1] W praktyce, aby policzyć wynik, najpierw zdejmuje się z planszy wszystkie martwe kamienie. Następnie wkłada się je w terytorium przeciwnika (każdy martwy kamień pomniejsza terytorium przeciwnika o jeden). Wygrywa gracz, któremu zostanie więcej wolnych przecięć terytorium. Uważny czytelnik sprawdzi, że wynik gry policzony w ten sposób jest równy wynikowi liczonemu metodą opisaną w głównym tekście.

1.5. Ko, czyli powtórzenie sytuacji na planszy

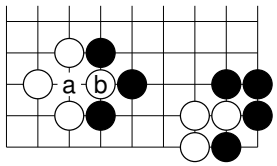


Diagram 1.6: Przykłady ko

W trakcie partii Go możliwe jest, że sytuacja na planszy się powtórzy. Sytuacje przedstawione na diagramie 1.6 nazywają się *ko*. Jeśli Czarne zagrają w *a*, zbiją biały kamień (*zbiją ko*). Następnie Białe będą mogły zagrać w Ⓑ i zbić dopiero co postawiony kamień Czarnych (*odbić ko*). Doprowadziłoby to do powtórzenia się sytuacji na planszy. Podobnie mogłaby się potoczyć rozgrywka w prawym dolnym narożniku.

Żeby zapobiec zapętleniu się gry, zasady Go zabraniają ruchu, który odbija ko, w następnym ruchu po zbiciu ko przez przeciwnika[2].

1.6. Walka ko - konsekwencja zasady ko

Po tym, jak gracz zbije ko, przeciwnik zmuszony jest do zagrania w innej części planszy. Wówczas gracz może *połączyć ko* (na przykład ruch Białych w *a* na diagramie 1.6). Mówi się wtedy, że gracz *wygrał ko*. Jeżeli jednak gracz również wykona ruch w innej części planszy, to w następnym ruchu przeciwnik będzie mógł odbić ko. Taka sytuacja nazywa się *walką ko*.

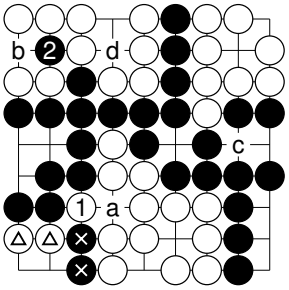


Diagram 1.7: Walka ko

Często zdarza się, że obu graczom bardzo zależy na wygraniu ko. Na diagramie 1.7 jeśli Białe wygrają ko w lewym dolnym narożniku, to *zabiją* (uczynią martwymi) dwa zaznaczone kamienie Czarnych. Z drugiej strony, wygranie ko przez Czarne zakończyłoby się śmiercią dwóch zaznaczonych kamieni Białych. Kiedy Białe biją ko ruchem ①, Czarne wykorzystują sprytną taktykę. Grają ruch w innej części planszy, który grozi, że zdobedą duży zysk. Ruch ❷ daje atari na dwa Białe kamienie. Białe, żeby ich nie stracić, muszą odpowiedzieć na *groźbę* ruchem w *b*. Wtedy Czarne odbiją ko ruchem w *a*. Nadejdzie kolej Białych na szukanie groźby. Ruch Białych w *c*, zagrozi dwóm czarnym kamieniom na prawej bandzie. Kiedy Czarne odpowiedzą, Białe ponownie odbiją ko. Czarne mają w zanadrzu jeszcze jedną groźbę na górnej bandzie (*d*), więc ostatecznie to Czarne wygrają ko.

Występowanie ko w znacznym stopniu komplikuje rozgrywkę w Go. Sprawia, że lokalne sytuacje, które mogły się wydawać odizolowane, stają się od siebie wzajemnie zależne.

[2]Bardzo rzadko w partiach Go pojawiają się też inne konfiguracje, w których pozycja na planszy mogłaby się powtórzyć. Różne zestawy reguł w różny sposób rozstrzygają takie sytuacje.

Podsumowanie

Partia Go rozgrywana jest przez dwóch graczy, którzy naprzemiennie kładą czarne i białe kamienie na przecięciach kwadratowej planszy. Celem każdego z graczy jest zdobycie kontroli nad większą częścią planszy. Jeśli gracz szczelnie otoczy łańcuch kamieni przeciwnika (zabierze mu wszystkie oddechy), to Go zbija. Zbite kamienie zostają zdjęte z planszy.

Kiedy gracze uznają, że plansza została już podzielona na teren Białych i Czarnych, pasują i przystępują do liczenia wyniku. Gracz otrzymuje punkt za każde otoczone pole oraz za każdy martwy kamień przeciwnika. Co ważne, żeby kamień przeciwnika został uznany za martwy, nie musi zostać zbity w trakcie partii. Jeśli pod koniec partii gracz ma możliwość kontynuowania gry i zbicia pewnego łańcucha kamieni przeciwnika, niezależnie od tego, jak przeciwnik by nie odpowiadał na jego ruchy, to kamienie tego łańcucha uznawane są za martwe.

W rozdziale 2 przedstawiam zagadnienia związane z końcową fazą gry. Do ich zrozumienia wystarcza sama znajomość reguł. Najważniejszy wniosek z niniejszego rozdziału, wymagany przy lekturze rozdziału 2, jest taki, że w trakcie partii *na zbiciu kamienia przeciwnika zyskuje się dwa punkty*: jeden punkt terytorium w miejscu po zbitym kamieniu oraz jeden punkt za zbity kamień. Podobnie, pod koniec partii *pola terytorium zajęte przez martwe kamienie przeciwnika liczy się dwukrotnie*.

Rozdział 2

Wartości ruchów w grze końcowej

Zwyczajowo wyróżnia się trzy fazy partii Go. W otwarciu (*fuseki*) gracze zarysowują swoje strefy wpływów i często wykonują standardowe rozegrania (*joseki*) w różnych rejonach planszy. Gra środkowa (*chuban*) rozpoczyna się wraz z pierwszą walką i toczy się, dopóki status wszystkich grup nie jest określony. Kiedy wiadomo już, które rejony planszy są pod czyją kontrolą, rozpoczyna się końcowa faza gry (*yose*).

W grze końcowej gracze wdają się już tylko w lokalne potyczki służące dokładnemu ustaleniu granic terytoriów. Te lokalne sytuacje często nie są ze sobą wzajemnie powiązane. Ponadto są one znacznie łatwiejsze do przeanalizowania niż całoplanszowe zmagania, którymi przepełniona jest gra środkowa. Pozycję na planszy można więc potraktować jako teoriomnogościową sumę mniejszych sytuacji. Dzięki tej charakterystyce końcówek możliwe jest stworzenie teoretycznego modelu określającego, w której z lokalnych sytuacji najbardziej opłaca się zagrać następny ruch.

W podrozdziale 2.1 wprowadzam podstawowe pojęcia teorii końcówek. W podrozdziale 2.2 formułuję klasyczny sposób liczenia wartości ruchów wypracowany przez goistów. W podrozdziale 2.3 przedstawiam inną wersję zasad Go, która pomaga zrozumieć, czym jest wartość ruchu. W podrozdziale 2.4 pokazuję, że końcowa faza gry jest skomplikowanym zagadnieniem. Przedstawiony matematyczny opis sytuacji występujących w grze końcowej nie wystarcza do zalgorytmizowania optymalnej gry. W podrozdziale 2.5 opisuję podstawy kombinatorycznej teorii gier. Pokazuję, że teoria ta pozwala na bardziej dokładną analizę gry końcowej niż klasyczna metoda używana przez goistów.

2.1. Wartość ruchu jako zmiana w lokalnym wyniku

W niniejszym podrozdziale wprowadzam podstawowe pojęcia teorii końcówek: lokalny wynik oraz wartość ruchu. Lokalne wyniki będę oznaczał wielką literą L z indeksami (C - Czarne i B - Białe) oznaczającymi, kto wykonywał kolejno zagrania w danej sytuacji. Przykładowo przez L_{CB} będę oznaczał lokalny wynik po ruchu Czarnych i odpowiedzi Białych. Wartość ruchu gracza będę oznaczał wielką literą W z indeksem C lub B.

Gra końcowa jako suma odizolowanych sytuacji

Na diagramie 2.1 terytoria graczy są w pełni ustalone za wyjątkiem kilku miejsc na planszy. Prawa część planszy jest kontrolowana przez Czarne, a lewa przez Białe. Kamień ⊗ jest martwy. Pośrodku planszy znajdują się cztery sytuacje, które można analizować osobno. Kluczowe pola w tych sytuacjach, na których ruch rozważałby każdy z graczy, są oznaczone jako a, b, c i d. Przyjrzyjmy im się po kolei.

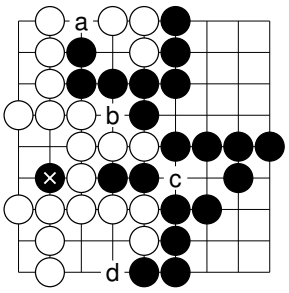


Diagram 2.1: Odizolowane lokalne sytuacje pod koniec partii

Ruch Czarnych w a zabiłby trzy kamienie Białych i otoczył jeden dodatkowy punkt terytorium. Białe zapobiegłyby temu, same grając w a.

Przecięcie b nie należy ani do terytorium Białych, ani Czarnych. Jest to tak zwane pole neutralne. Niezależnie kto tam zagra, nie zdobędzie żadnego punktu

W sytuacji poniżej może się wydawać, że Czarne otaczają dwa punkty terytorium. Jednakże dwa kamienie Czarnych stoją pod atari i Białe mogą je zbić ruchem w c. Jeśli zagrają tam Czarne, to uratują swoje kamienie.

To, czy Białe otoczą trzypunktowe terytorium na dolnej bandzie, zależy od tego, który z graczy zagra w d.

Zagrania w pozostałych częściach planszy są nierozsądne. Ruchy wewnątrz własnego terytorium zmniejszają liczbę własnych otoczonych przecięć. Ruchy wewnątrz terytorium przeciwnika tylko dostawiają martwe kamienie, powiększając wynik przeciwnika.

Chcielibyśmy wypracować metodę, która powie nam, który ruch na planszy jest najwięcej wart. W tym celu wprowadzimy najpierw pojęcie lokalnego wyniku, które pozwala każdą sytuację analizować osobno.

Lokalny wynik i wartość ruchu

Definicja 2.1.1. *Lokalny wynik jest to spodziewana różnica w lokalnej liczbie punktów Czarnych i Białych.*

Definicja ta nie jest precyzyjna. Jej sprecyzowanie stanie się możliwe dopiero w podrozdziale 2.2.

W przypadku lokalnej sytuacji, która jest w pełni wyjaśniona, definicja jest jasna. Przykładowo, na diagramie 2.1 w lewym dolnym narożniku lokalny wynik to 2 punkty na korzyść Białych. Jeśli Czarne wykonają ruch c, to lokalny wynik na środku będzie równy 1 punkt na korzyść Czarnych. Lokalne wyniki będziemy zawsze przedstawiać z perspektywy Czarnych. Jeżeli lokalny wynik będzie korzystny dla Białych, powiemy, że $L < 0$. Na przykład w lewym dolnym narożniku $L = -2$.

Lokalny wynik w sytuacji, która nie jest jeszcze w pełni wyjaśniona, określa się jako średnią arytmetyczną lokalnego wyniku po ruchu Czarnych i lokalnego wyniku po ruchu Białych:

$$L = \frac{L_C + L_B}{2} \tag{2.1}$$

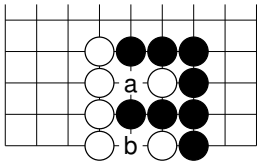


Diagram 2.2: Lokalny wynik w niedogranej pozycji

Diagram 2.2 pomaga zrozumieć wzór 2.1. Przedstawione są dwie identyczne lokalne sytuacje. Przyjmijmy, że w jednej z nich lokalny wynik jest równy L. Potraktujmy następnie obie sytuacje jako jedną większą lokalną sytuację. W tej większej lokalny wynik jest równy $2L$.

Jak zobaczymy, ta większa sytuacja jest już rozstrzygnięta. Jeśli Czarne zagrają w a, to Białe odpowiedzą w b, i na odwrót. Niezależnie, który z graczy wykona ruch jako pierwszy, Czarne zbiją jeden kamień, a Białe *połączą* drugi. Sytuacja zakończy się wynikiem 2 punktów dla Czarnych, a więc $2L = 2$, czyli $L = 1$.

Definicja 2.1.2. *Wartość ruchu jest to zmiana*[1] *w lokalnym wyniku spowodowana przez ruch. Wartość ruchu jest zawsze przedstawiana z perspektywy grającego ten ruch. Wartość ruchu Czarnych to* $W_C = L_C - L$. *Wartość ruchu Białych to* $W_B = L - L_B$.

W definicji wartości ruchu Białych odejmuje się L_B od L dlatego, że przyjęliśmy konwencję, zgodnie z którą lokalne wyniki są przedstawiane z perspektywy Czarnych. Dlatego zachodzi $L_B \leq L$ i wartość ruchu jest nieujemna. Zauważmy, że wartości ruchów dla obu graczy są równe:

$$W_B = L - L_B = \frac{L_B + L_C}{2} - L_B = \frac{L_C - L_B}{2} = L_C - \frac{L_B + L_C}{2} = L_C - L = W_C \tag{2.2}$$

W praktyce wartość ruchu najłatwiej jest obliczyć, używając środkowego wyrażenia w powyższym ciągu równości:

$$W_B = W_C = \frac{L_C - L_B}{2} \tag{2.3}$$

[1] Często przyjmowana jest inna konwencja, zgodnie z którą wartość ruchu to różnica pomiędzy lokalnym wynikiem po ruchu Czarnych a lokalnym wynikiem po ruchu Białych: $W_C = L_C - L_B\,(= W_B)$. Autor pracy uważa tę konwencję za bardziej naturalną i to jej poleca używać graczom Go. Jednakże w świetle rozważań przedstawionych w podrozdziale 2.5, to konwencja z głównego tekstu pozwala na precyzyjne ujęcie bardziej szczegółowych zagadnień.

Zależności pomiędzy lokalnymi wynikami i wartościami ruchów ilustruje diagram 2.3. Przykłady od (*a*) do (*d*) odpowiadają lokalnym sytuacjom z diagramu 2.1.

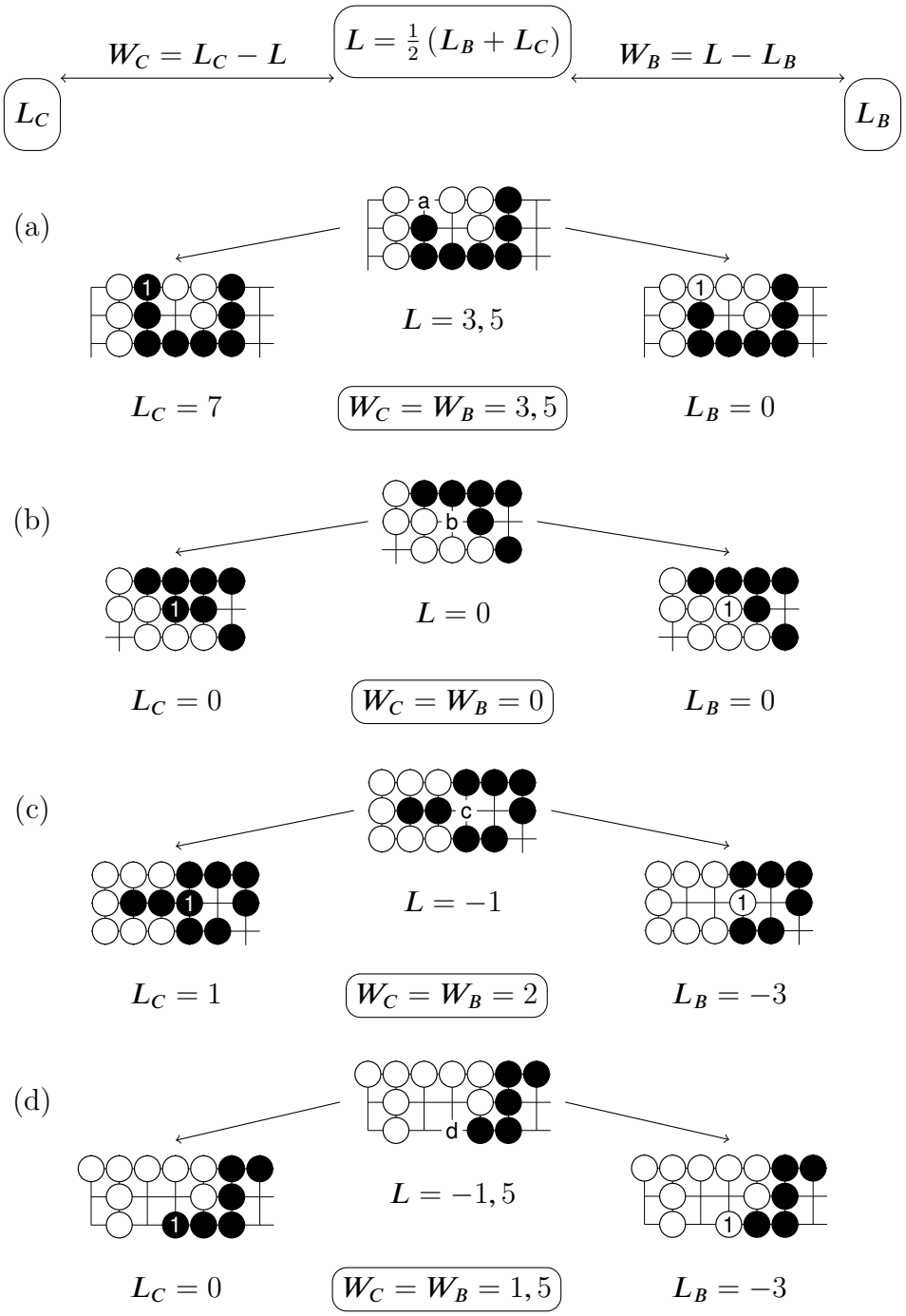


Diagram 2.3: Lokalne wyniki i wartości ruchów

Uważny Czytelnik sprawdzi, że w przykładzie (*d*) po ruchu Czarnych wszystkie trzy niezajęte pola są neutralne. Najtrudniejsza w analizie może być sytuacja po ruchu Białych w przykładzie (*c*). Białe posiadają w tej sytuacji dwa punkty ze względu na zbite kamienie oraz jeden punkt terytorium (a nie dwa, jak mogłoby się wydawać na pierwszy rzut oka). Pole na lewo od kamienia ① nie jest punktem terytorium, ponieważ kiedy Czarne dadzą atari, stawiając na prawo od ①, Białe będą musiały się połączyć.

Granice terytoriów w prawdziwej grze

W prawdziwych partiach granice lokalnych sytuacji rzadko kiedy są tak jasno określone jak na diagramie 2.1. W pozycji przedstawionej na diagramie 2.4 można się zastanawiać, które przecięcia zaliczyć do lokalnych sytuacji na bandach i w środku planszy. W praktyce taką decyzję podejmuje się według uznania. Na przykład w sytuacji na prawej bandzie (związanej z ruchem b) jako punkty możliwe do zdobycia przez Czarne można liczyć wszystkie przecięcia oznaczone jako $\times$. Do lokalnej sytuacji na dolnej bandzie (związanej z ruchem c) można zaliczyć wszystkie przecięcia $\triangle$. Doświadczony gracz widzi jednak, że nie musi brać pod uwagę wszystkich tych przecięć. Niektóre z nich we wszystkich wariantach wartych rozważenia i tak będą ostatecznie należały do terytorium Czarnych. Dlatego przy analizie sytuacji może ograniczyć swoją uwagę do mniejszego obszaru.

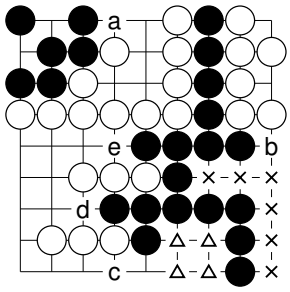


Diagram 2.4: Bardziej realistyczna pozycja z gry końcowej

Warto zaznaczyć, że celem analizy lokalnych sytuacji jest ustalenie wartości ruchów. Lokalne wyniki są jedynie narzędziem pomocniczym. Wartość ruchu jest zdefiniowana jako różnica pomiędzy lokalnymi wynikami po ruchu i przed ruchem. Dlatego doświadczony gracz, który zaliczy do lokalnej sytuacji mniej spośród zaznaczonych przecięć, otrzyma na końcu takie same wartości ruchów.

Stwierdzenie 2.1.3. *Jeśli przy analizie sytuacji do każdego z lokalnych wyników doda się pewną liczbę $r \in \mathbb{R}$, to obliczone przy ich pomocy wartości ruchów pozostaną bez zmian.*

Przykładowo, doświadczony gracz wie, że na dolnej bandzie trzy spośród przecięć $\triangle$ w każdym wartym rozważenia wariancie skończą jako punkty Czarnych. Dlatego jako należące do lokalnej sytuacji uzna tylko jedno przecięcie $\triangle$. Każdy z obliczonych przez niego lokalnych wyników będzie po prostu mniejszy o 3, w szczególności $L' = L - 3$ oraz $L'_C = L_C - 3$. Obliczona przez niego wartość ruchu będzie równa

$$W'_C = L'_C - L' = (L_C - 3) - (L - 3) = L_C - L = W_C$$

Stwierdzenie 2.1.3 przyda się w podrozdziale 2.5. Będziemy tam bowiem przyjmować wygodną konwencję, zgodnie z którą w każdej lokalnej sytuacji obecny wynik jest remisowy $L = 0$.

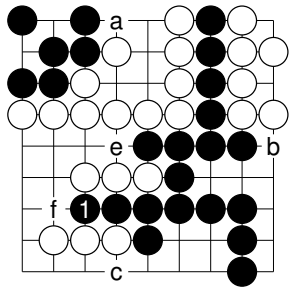


Diagram 2.5: Groźba wtargnięcia w terytorium

O ile kwestia terytoriów Czarnych była w miarę jasna, o tyle uważny Czytelnik może być zaniepokojony kwestią terytorium Białych na lewej bandzie. Czarne mogą próbować wkraść się w to terytorium w trzech różnych miejscach (ruchy c, d, e na diagramie 2.4). Formalnie rzecz biorąc, sytuacje te nie są odizolowane. Spójrzmy, co się dzieje, kiedy Czarne zagrają w d, czyli w ❶ na diagramie 2.5. Białe teoretycznie mogłyby rozważyć ruch w każdej dostępnej lokalnej sytuacji (np. na polu a, b, e albo polu położonym na prawo od c). Wówczas jednak Czarne kontynuowałyby ruchem w f, całkowicie niszcząc terytorium Białych. Pozostałoby to nie bez wpływu na wartości ruchów w pozostałych lokalnych sytuacjach.

W praktyce doświadczony gracz od razu odpowie na ruch ❶, blokując swoje terytorium ruchem w f. Analizując pozycje gry końcowej przyjmuje się założenie, że gracze posiadają ten element wiedzy eksperckiej. Wówczas sytuacje związane z terytorium Białych można traktować jako odizolowane, a punkty na lewej bandzie - jako

pewne terytorium Białych. *Wymiana* ❶-*f* nazywa się *przywilejem* Czarnych dlatego, że mogą one zagrać ruch ❶ w dowolnym momencie i mają pewność, że Białe na niego odpowiedzą.

Teoria przedstawiona w niniejszym podrozdziale nie wystarcza jeszcze do poprawnego analizowania sytuacji w grze końcowej. Następny podrozdział zaczynam od wykazania słabości tej teorii. Dalej przedstawiam klasyczny sposób liczenia wartości ruchów. Precyzuję również kwestię wymian stanowiących przywilej jednego z graczy.

2.2. Klasyczny sposób liczenia wartości ruchów

W poprzednim podrozdziale wartość ruchu została zdefiniowana jako zmiana w lokalnym wyniku. Lokalny wynik w każdej niewyjaśnionej sytuacji został określony jako średnia arytmetyczna lokalnych wyników w sytuacji po ruchu Czarnych oraz w sytuacji po ruchu Białych. Niedługo okaże się, że określenie to wymaga pewnej rewizji.

W niniejszym podrozdziale będziemy rozważać sekwencje złożone z dwóch ruchów. Drugi ruch w sekwencji nazywa się *kontynuacją*, jeżeli jest on zagrany przez tego samego gracza co pierwszy ruch. W przeciwnym przypadku drugi ruch w sekwencji nazywa się *odpowiedzią* przeciwnika. Lokalną sekwencję złożoną z ruchu i odpowiedzi przeciwnika nazywa się *wymianą*. Wartość drugiego ruchu będziemy oznaczać wielką literą W z dwoma indeksami. Na przykład W_{BC} to wartość odpowiedzi Czarnych na ruch Białych. Wprowadzimy również pojęcie *pewnego zysku* gracza. Pewny zysk będziemy oznaczali wielką literą Z z indeksem C lub B.

Wartość ruchu z kontynuacją

Będziemy rozważać sytuacje, które nie są w pełni wyjaśnione po tylko jednym zagraniu. Na przykład na diagramie 2.6 po ruchu Czarnych w *a* granice terytoriów wciąż nie są ustalone. Analizę takich sytuacji znacząco upraszcza pojęcie pewnego zysku. Nazwa bierze się stąd, że jest to zysk, który gracz odniesie z ruchu na pewno - nawet jeśli przeciwnik odpowie na ruch.

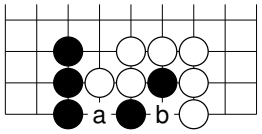


Diagram 2.6: Ruch Czarnych *a* ma kontynuację w *b*.

Definicja 2.2.1. *Pewny zysk gracza jest to różnica pomiędzy lokalnym wynikiem po ruchu przeciwnika a lokalnym wynikiem po ruchu gracza i odpowiedzi przeciwnika. Pewny zysk gracza jest zawsze przedstawiany z jego perspektywy. Pewny zysk Czarnych to* $Z_C = L_{CB} - L_B$. *Pewny zysk Białych to* $Z_B = L_C - L_{BC}$.

Pewne zyski, podobnie jak wartości ruchów, przedstawiane są z perspektywy grającego i dlatego są zawsze nieujemne. Lokalne wyniki są przedstawiane z perspektywy Czarnych i zachodzi $L_C > L_{BC}$. Dlatego pewny zysk Białych określamy, odejmując L_{BC} od L_C.

Dla ustalenia uwagi będziemy zakładać, że ruch, który posiada kontynuację, należy do Czarnych (tak jak na diagramie 2.6). Wartość odpowiedzi Białych na ruch Czarnych jest z definicji równa $W_{CB} = L_C - L_{CB}$, a zatem $L_C = L_{CB} + W_{CB}$. Z definicji pewnego zysku otrzymujemy za to $L_B = L_{CB} - Z_C$. Podstawmy te dwie równości do wzoru 2.3. Uzyskujemy nową postać wzoru na wartość ruchu, która w praktyce jest łatwiejsza w użyciu.

$$W_C = \frac{1}{2}(L_C - L_B) = \frac{1}{2}(L_{CB} + W_{CB} - (L_{CB} - Z_C)) = \frac{1}{2}(Z_C + W_{CB}) \tag{2.4}$$

Przyjrzyjmy się sytuacji z diagramu 2.6. Diagram 2.7 przedstawia drzewo gry dla tej sytuacji oraz ilustruje ogólne zależności pomiędzy pewnym zyskiem, lokalnymi wynikami oraz wartościami ruchów. Żeby obliczyć lokalne wyniki w węzłach takiego drzewa, trzeba zacząć od liści, w których sytuacje są wyjaśnione. Następnie, idąc ku górze drzewa, wyliczamy lokalny wynik w każdym węźle jako średnią arytmetyczną wyników w jego dzieciach.

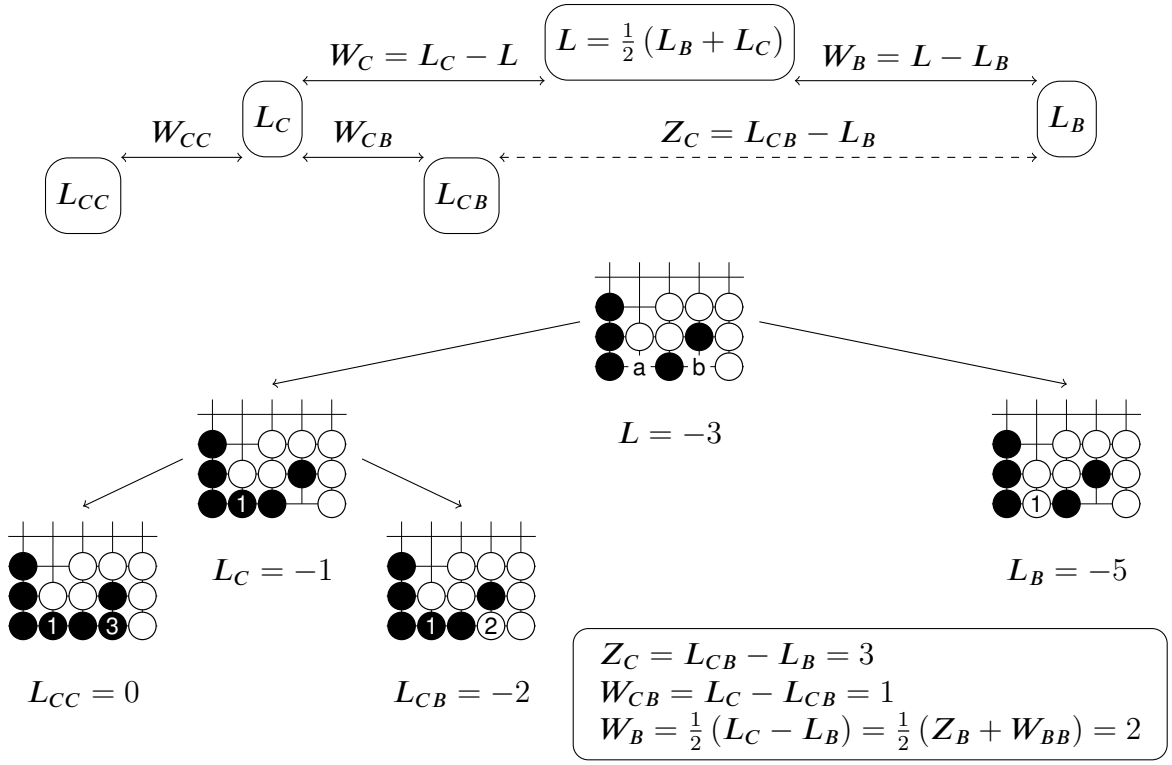


Diagram 2.7: Pewny zysk a wartość ruchu z kontynuacją

Przywilej gracza, czyli sente

Spróbujmy policzyć wartości ruchów Białych na diagramie 2.8.

Na lewej bandzie sytuacje po ruchu każdego z graczy w a są wyjaśnione.

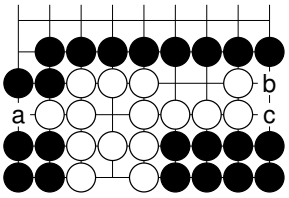


Diagram 2.8: Duży ruch i ruch z dużą kontynuacją

$$L_C = 0$$
$$L_B = -8$$
$$W_B = \frac{L_C - L_B}{2} = \frac{8}{2} = 4$$

Na prawej bandzie do policzenia wartości ruchu w b spróbujemy użyć wzoru 2.4. Wynik, który uzyskamy, pokaże, że wypracowana w poprzednim podrozdziale metoda liczenia wartości ruchów nie zawsze się sprawdza.

$$L_C = -17$$
$$L_{CB} = -16$$
$$Z_C = L_{CB} - L_B = -16 - (-17) = 1$$
$$L_{CC} = 0$$
$$W_{CC} = \frac{1}{2}(L_{CC} - L_{CB}) = \frac{1}{2}(0 - (-16)) = 8$$
$$W_B = \frac{1}{2}(1 + 8) = 4,5$$

Wygląda na to, że ruch b jest więcej wart niż a. Taki wniosek wyda się dziwny dla każdego doświadczonego gracza. Czy Białe w przedstawionej sytuacji powinny wybrać ruch w b i pozwolić Czarnym uratować kamienie w lewym dolnym narożniku? Zamiast tego Białe mogłyby przecież zbić kamienie po lewej stronie, a po ruchu Czarnych w b zbić też kamienie po prawej stronie ruchem w c. Ten rezultat wygląda lepiej dla Białych. Przyjrzyjmy się jeszcze raz stosowanym definicjom.

Zgodnie ze wzorem 2.1 lokalny wynik w sytuacji po prawej stronie jest równy

$$L = \frac{1}{2}((-8) + (-17)) = -12,5$$

Załóżmy jednak, że po ruchu Czarnych w b, Białe zawsze odpowiadają w c. Po takiej wymianie lokalny wynik jest równy -16. Jeśli tylko Białe przyjmą opisaną taktykę, lokalny wynik będzie dla nich bardziej korzystny niż wynikałoby to ze wzoru. Jest to paradoks. Nie można mówić, że spodziewana różnica punktowa (p. definicja 2.1.1) w tej sytuacji to $-12,5$, skoro Białe przyjmując prostą taktykę, mogą zawsze liczyć na lepszy wynik.

Jeżeli lokalny wynik po wymianie jest bardziej korzystny dla drugiego gracza niż wynik obliczony ze wzoru 2.1, to taką wymianę nazywa się przywilejem pierwszego gracza. Pierwszy ruch takiej wymiany nazywa się *sente*. W takich sytuacjach uznaje się, że lokalny wynik jest równy lokalnemu wynikowi po dokonaniu wymiany.

Lemat 2.2.2. *Wymiana stanowi przywilej gracza wtedy i tylko wtedy, gdy wartość kontynuacji przekracza pewny zysk.*

Dowód. Dla ustalenia uwagi zakładamy, że pierwszy ruch wymiany należy do Czarnych. Lokalny wynik po wymianie jest bardziej korzystny dla Białych wtedy, gdy $L > L_{CB}$. Z definicji pewnego zysku otrzymujemy:

$$\begin{aligned} Z_C &= L_{CB} - L_B \\ L_B &= L_{CB} - Z_C \end{aligned} \tag{2.5}$$

Ze wzoru na wartość ruchu 2.3 otrzymujemy:

$$\begin{aligned} W_{CC} &= \frac{L_{CC} - L_{CB}}{2} \\ W_{CC} + L_{CB} &= \frac{L_{CC} + L_{CB}}{2} \end{aligned} \tag{2.6}$$

Sytuacja po ruchu Czarnych nie jest wyjaśniona. Zgodnie ze wzorem 2.1:

$$\begin{aligned} L_C &= \frac{(L_{CC} + L_{CB})}{2} \\ L = \frac{1}{2}(L_C + L_B) &= \frac{1}{2}\left(L_B + \frac{L_{CC} + L_{CB}}{2}\right) \end{aligned} \tag{2.7}$$

Podstawmy równości 2.5 i 2.6 do równości 2.7

$$\begin{aligned} L &= \frac{1}{2}\left(L_B + \frac{L_{CC} + L_{CB}}{2}\right) \\ L &= \frac{1}{2}(L_{CB} - Z_C + W_{CC} + L_{CB}) \\ L - L_{CB} &= \frac{1}{2}(W_{CC} - Z_C) \end{aligned}$$

Zatem nierówność $L > L_{CB}$ jest równoważna nierówności $W_{CC} > Z_C$. □

Na omawianą sytuację można też spojrzeć w inny sposób. W fazie końcowej gracze wykonują ruchy o coraz niższej wartości. Mówi się, że *temperatura* (czyli wartość największego dostępnego ruchu) spada. Załóżmy, że Czarne mają dostępny ruch, dla którego pewny zysk jest równy Z_C, a kontynuacja jest równa $W_{CC} > Z_C$. Ruch Białych w tamtym miejscu poprawiłby lokalny wynik na korzyść Białych o $L_B - L_{CB} = Z_C$. Dlatego Białym nie opłaca się tam zagrać, dopóki temperatura na planszy przewyższa Z_C. Jednak Czarne zagrają tam, kiedy tylko temperatura na planszy spadnie poniżej W_{CC}. Wówczas, po ruchu Czarnych, największym ruchem dostępnym Białym będzie odpowiedź. Dlatego spodziewamy się, że zanim zagrają tam Białe, to zdążą tam zagrać Czarne i Białe odpowiedzą. Właśnie dlatego *spodziewana różnica punktowa* (p. definicja 2.1.1) w takiej sytuacji jest równa lokalnemu wynikowi po dokonaniu wymiany.

Algorytm liczenia wartości ruchów

Powyższa analiza pozwala na precyzyjne sformułowanie klasycznego sposobu liczenia lokalnego wyniku oraz wartości ruchów.

Jeśli w lokalnej sytuacji granice terytoriów są ustalone, to lokalny wynik określa się jako różnicę w lokalnej liczbie punktów Czarnych i Białych. Jeśli w lokalnej sytuacji dokonanie pewnej wymiany jest przywilejem jednego z graczy, wówczas lokalny wynik określa się jako lokalny wynik po dokonaniu tej wymiany (na przykład $L = L_{CB}$, jeśli pierwszy ruch wymiany należy do Czarnych). W pozostałych przypadkach lokalny wynik określa się jako średnią arytmetyczną pomiędzy lokalnym wynikiem po ruchu Czarnych a lokalnym wynikiem po ruchu Białych.

Żeby sprawdzić, czy wymiana stanowi przywilej, należy policzyć pewny zysk, czyli różnicę pomiędzy sytuacją po dokonaniu wymiany przez pierwszego gracza a sytuacją po ruchu drugiego gracza. Jeżeli różnica ta jest mniejsza niż wartość kontynuacji ruchu pierwszego gracza, to wymiana stanowi przywilej.

Wartość ruchu jest określona jako zmiana w lokalnym wyniku spowodowana przez ruch.

W sytuacji, w której żaden z graczy lokalnie nie posiada ruchu sente, wartości ruchów obu graczy są sobie równe na mocy równości 2.2. Zachodzi wówczas:

$$W_B = W_C = \frac{W_C - W_B}{2}$$

W przeciwnym przypadku wartość ruchu dla obu graczy się różni. Ruch zapobiegający zagraniu sente jest wart tyle co pewny zysk przeciwnika, ponieważ zmiana w lokalnym wyniku to odebranie przeciwnikowi pewnego zysku. Na przykład, jeśli jest to ruch Białych, to posiada on wartość:

$$W_B = L - L_B = L_{CB} - L_B = Z_C$$

Natomiast ruch w sente jest wart tyle, co jego kontynuacja:

$$W_C = L_C - L = L_C - L_{CB} = W_{CB} = W_{CC}$$

W praktyce gracze Go nie używają pojęcia wartości ruchu w odniesieniu do ruchów sente. Gracz, wykonując ruch sente, i tak nie spodziewa się zagrać jego kontynuacji. Przeciwnik od razu odpowie na ruch i sytuacja się ustabilizuje bez zmiany lokalnego wyniku. Goistom w odniesieniu do ruchów sente wystarcza informacja, że kontynuacja jest względnie duża.

Powyższy opis streszcza całą klasyczną teorię liczenia wartości ruchów. Użyte przykłady miały nieduży stopień skomplikowania. Za każdym razem po ruchu jednego z graczy sytuacja była w pełni wyjaśniona. Natomiast po ruchu drugiego gracza do wyjaśnienia sytuacji brakowało co najwyżej jednego ruchu więcej. Mimo to omawiane przykłady pozwoliły na wprowadzenie wszystkich kluczowych pojęć. Są one wystarczające do opisu również bardziej skomplikowanych pozycji. Dalsza część niniejszego rozdziału służy rozważeniu, do czego przydaje się przedstawiona teoria. Podrozdział 2.5 pokazuje też, jak dzięki kombinatorycznej teorii gier można dokonać jeszcze dokładniejszej analizy sytuacji pojawiających się w grze końcowej.

2.3. Ekonomiczna wersja Go

Uważny Czytelnik może się zastanawiać, czym są wartości ruchów, które wyliczaliśmy w poprzednim podrozdziale. Co oznacza, że ruch jest wart $2,5$ punktu, jeśli pod koniec partii terytoria obu graczy to liczby całkowite? Pewnej odpowiedzi na to pytanie dostarcza artykuł Elwyna Berlekampa *The Economist's View of Combinatorial Games* [2]. W artykule autor przedstawia modyfikację zasad Go, a także innych gier badanych przez kombinatoryczną teorię gier. W zaproponowanej *ekonomicznej* wersji zasad na początku partii obaj gracze dysponują pewnym kapitałem i celem gry jest powiększenie swojego kapitału. Do gry zostaje dodany element licytacji i uiszczania opłat za wykonywane ruchy. Autor pokazuje, że wartość ruchu obliczona zgodnie z klasyczną metodą stanowi najwyższą kwotę, którą racjonalny gracz może zapłacić za prawo do jego wykonania.

Opłata za ruch

Partia ekonomicznego Go rozpoczyna się od pewnej ustalonej pozycji z fazy końcowej. Za wykonanie każdego ruchu trzeba zapłacić przeciwnikowi pewną kwotę. Kwota ta jest ustalana przez graczy co kilka ruchów. Kiedy partia dobiega końca, następuje ostatnia opłata: gracz, który osiągnął niższy wynik, musi zapłacić przeciwnikowi kwotę równą różnicy punktowej. Gracz, który kończy z wyższym kapitałem niż przed rozpoczęciem gry, zostaje zwycięzcą.

W partii oprócz samego grania ruchów mają miejsce również licytacje:

- jedna licytacja komi na początku partii
- wiele licytacji wysokości opłaty za ruch, która będzie obowiązywać przez najbliższych kilka ruchów

Na początku partii trzeba ustalić kolory graczy oraz wysokość komi, które Czarne będą musiały zapłacić Białym. Odbywa się to w drodze licytacji. Jeśli jeden z graczy zaproponuje wyższe komi, to będzie grał Czarnymi. Jeśli obaj gracze poprawnie wycenią sytuację na planszy i zaproponują optymalne komi, to kolory zostaną wylosowane. Następnie przy poprawnej grze obu graczy partia powinna zakończyć się remisem.

Podczas partii gracze deklarują, jaką kwotę są skłonni zapłacić za prawo do wykonania następnego ruchu. Wyższa z zaproponowanych kwot zostaje aktualną opłatą za ruch. Prawo do wykonania ruchu uzyskuje gracz, który zgłosił tę kwotę. W przypadku zgłoszenia tych samych kwot decyzję o tym, kto ma wykonywać następny ruch, podejmuje sędzia.

Kiedy wysokość opłaty została ustalona, gracze naprzemiennie wykonują ruchy i za każdy uiszczają opłatę. Dozwolone jest również spasowanie, za które nie trzeba nic płacić przeciwnikowi. Kiedy obaj gracze spasują, następuje nowa licytacja wysokości opłaty za ruch. Po ustaleniu nowej kwoty partia zostaje wznowiona do czasu dwóch pasów, po których następuje kolejna licytacja. Kiedy gracze nie są już skłonni do licytowania dodatnich kwot, partia dobiega końca. Wówczas następuje ostatnia opłata wyrównująca różnicę w terytoriach i zbitych kamieniach i ustalony zostaje zwycięzca.

Strategia książkowa

Rozgrywka w ekonomicznym Go może mieć wyraźnie różny przebieg od zwykłego Go. Może się zdarzyć, że jeden gracz wykonuje na planszy kilka ruchów z rzędu. Ponadto, ważnym elementem gry jest umiejętność zaproponowania odpowiedniej wysokości opłaty za ruch.

Berlekamp w swoim artykule dowodzi, że przy każdej licytacji gracz powinien proponować kwotę równą wartości największego z dostępnych mu ruchów. Proponowanie kwoty wyższej na nic się nie zda. Jeśli zostanie ona przyjęta, to gracz na każdym dostępnym ruchu tylko by tracił i jedyną rozsądną decyzją będzie spasowanie. Jeśli natomiast gracz zalicytuje kwotę niższą, to przeciwnik ma szansę to wykorzystać, licytując kwotę minimalnie wyższą. To przeciwnik będzie wykonywał następny ruch i nie jest wykluczone, że na tym zyska.

Następnie Berlekamp przedstawia prostą strategię wybierania ruchów. Dowodzi, że jeśli gracz proponuje poprawne kwoty w trakcie licytacji, to strategia ta jest nieprzegrywająca. Nazywa ją strategią książkową[2]:

1. Jeżeli w lokalnej sytuacji, w której dopiero co zagrał przeciwnik, wartość ruchu przekracza wysokość opłaty za ruch, odpowiedz na ruch przeciwnika (czyli w praktyce: odpowiadaj na ruchy sente).

2. W przeciwnym przypadku, jeżeli któryś z dostępnych ruchów ma wartość nie mniejszą niż opłata za ruch, zagraj ruch o największej wartości.

3. W przeciwnym przypadku spasuj.

[2]Uważny Czytelnik może być zaskokoczony pierwszym punktem w przedstawionej strategii. Można się zastanawiać, czy nie wystarczyłoby po prostu wybierać ruchów o największej wartości, o ile tylko ich wartość przekracza opłatę za ruch. Omówienie tej kwestii zanjduje się w dodatku A. Przed przystąpieniem do lektury tego dodatku, warto przeczytać podrozdział 2.4, aby zapoznać się z zagadnieniem *tedomari*.

2.4. Problemy teorii końcówek

Podrozdział 2.2 pokazał sposób przyporządkowania lokalnym sytuacjom liczb nazywanych lokalnymi wynikami. Pociągnęło to za sobą również przyporządkowanie ruchom liczb nazywanych ich wartościami. Wiemy, że w ekonomicznej wersji zasad Go nieprzegrywająca strategia polega na wybieraniu ruchów o najwyższych wartościach i odpowiadaniu na ruchy sente przeciwnika. Co to jednak oznacza dla zwykłego Go? Czy w fazie końcowej wystarczy za każdym razem wybierać zagranie o najwyższej wartości? Okazuje się, że sprawa nie jest taka prosta.

W niniejszym podrozdziale pokażemy przykłady pozycji, w których wypracowana do tej pory teoria nie wystarcza do znalezienia najlepszego ruchu. Omówimy również trudności, które napotykają próby podzielenia planszy na odizolowane lokalne sytuacje.

Ruchy o tej samej wartości

Dzięki klasycznej metodzie wiemy, jak obliczyć wartości ruchów. Co jednak jeśli na planszy znajduje się kilka ruchów o tej samej wartości? Czy można wybrać dowolny z nich? Jak się okaże, czasami niepoprawny wybór spośród takich zagrań może doprowadzić do straty punktowej.

Diagram 2.9 przedstawia trzy ruchy, które dla Czarnych mają wartość jednego punktu. Nietrudno przestudiować wszystkie możliwe warianty w tej sytuacji. Czarne powinny zacząć od ruchu *a*. Wówczas partia zakończy się remisem. Jeśli rozpoczną od ruchu *b*, Białe dokonają wymiany na górnej bandzie, po czym zagrają w *c*. Czarne przegrają o jeden punkt.

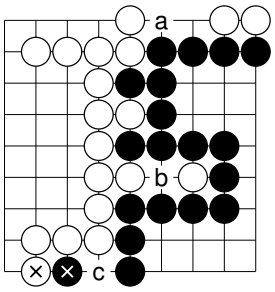


Diagram 2.9: Wybór spośród jednopunktowych ruchów

Wyobraźmy sobie jednak, że zaznaczone kamienie na dolnej bandzie są przesunięte o jeden w prawo, to znaczy kamień Czarnych ⊗ stoi na przecięciu *c*, a kamień Białych ⊗ stoi w miejscu kamienia Czarnych ⊗. W takiej pozycji żeby uzyskać remis, Czarne powinny zacząć od ruchu *b*. Zagranie Czarnych w *a* doprowadzi do porażki.

W przedstawionym przykładzie samo obliczenie wartości ruchów nie wystarcza do znalezienia optymalnego zagrania. Przykład dowodzi jednak czegoś więcej. Do wyboru pomiędzy ruchami *a* i *b* nie wystarcza lokalna analiza tych dwóch sytuacji. Inne zagranie jest optymalne w zależności od globalnej sytuacji na planszy. Wynika stąd, że niemożliwe jest stworzenie funkcji, która wystarczyłaby do znalezienia optymalnego wariantu poprzez zwykłe wybieranie ruchu o największej wartości.

Tedomari, czyli ostatni ruch

Możliwe są również sytuacje, w których poprawną decyzją jest wybranie ruchu o mniejszej wartości. Na diagramie 2.10 ruch na górnej bandzie ② jest wart 6,5 punktu, a ruch na dolnej bandzie ❶ zaledwie 5 punktów. Jeśli jednak Czarne zagrałyby swój pierwszy ruch na górze, w miejscu kamienia ②, Białe zagrałyby w miejscu kamienia ❶ i wygrały partię o jeden punkt. Tymczasem wariant pokazany na diagramie zapewnia Czarnym zwycięstwo o dwa punkty.

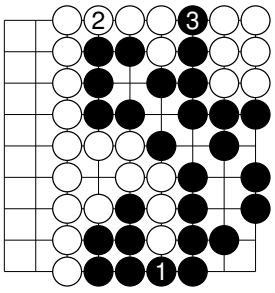


Diagram 2.10: Ruch o mniejszej wartości zapewnia zwycięstwo

Nietrudno zauważyć, na czym polega ta paradoksalna sytuacja. Po dwóch ruchach ❶ i ② suma lokalnych wyników zmieniła się w niekorzystny sposób dla Czarnych. Jednakże to Czarne zagrały następny ruch ❸, który był zarazem ostatnim ruchem partii. Koncept ostatniego spośród kilku dużych ruchów nazywa się *tedomari*. Czasami opłaca się wybrać ruch, który nie ma najwyższej wartości, po to, żeby zdobyć tedomari.

W prawdziwych partiach nie zdarza się, aby ostatnie ruchy dostępne na planszy miały tak dużą wartość, jak to ma miejsce na diagramie 2.10. Zazwyczaj po zagraniu ruchów wartych około 5 punktów na planszy pozostaje jeszcze wiele ruchów o niższych wartościach. Wówczas chwilowa przewaga, którą Czarne uzyskają dzięki sekwencji ❶-②-❸ zostaje wkrótce zniwelowana, ponieważ Białe zagrają następny ruch, który również będzie posiadał pewną wartość. Dlatego w praktyce rzadko kiedy na zdobyciu tedomari można zyskać tak dużo jak na pokazanym przykładzie. Tedomari pozostaje jednak kwestią, o której gracz powinien pamiętać w końcowej fazie gry.

Zależności pomiędzy lokalnymi sytuacjami

Przedstawiona do tej pory teoria bazuje na założeniu, że pozycję na planszy można podzielić na kilka odizolowanych od siebie sytuacji. Jednakże takie założenie nie zawsze się sprawdza.

W przypadku opisanych w podrozdziale 1.6 walk ko wszystkie sytuacje na planszy mogą mieć na siebie wzajemny wpływ. Jeżeli gdzieś na planszy szykuje się duża walka ko, gracz może uzależnić od tego swoje decyzje podejmowane w innych rejonach planszy. Czasami powinien wybrać wariant, który kończy się dla niego stratą punktową, ale za to pozostawia mu wiele gróźb do ko.

Również w przypadku braku walk ko podział planszy na lokalne sytuacje może być problematyczny. Diagram 2.11 pokazuje pozycję, w której dwie pozornie odizolowane sytuacje okazują się być od siebie wzajemnie zależne. Doświadczony gracz rozpozna sekwencję ①-❷-③ jako częstą technikę używaną w grze końcowej. Po ruchu Białych ③ Czarne w większości przypadków łączą swoje kamienie ruchem w *a*. Jeśli Czarne tam nie zagrają, kontynuacja Białych może doprowadzić do destrukcji terytorium Czarnych.

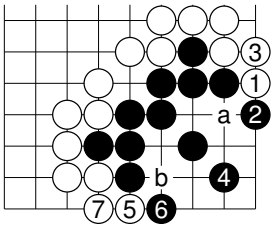


Diagram 2.11: Ruch ❹ zabezpiecza dwie słabości naraz

Okazuje się jednak, że po ruchu Czarnych ❹ terytorium Czarnych jest już bezpieczne. Jest to równie dobre zabezpieczenie jak łączenie w *a*. Następnie Białe używają analogicznej techniki na dolnej bandzie, grając sekwencję ⑤-❻-⑦. W tej chwili Czarne nie muszą się łączyć w *b*, ponieważ ich kształt jest już zabezpieczony przez kamień ❹. Mniej doświadczony gracz w przedstawionej sytuacji nie zauważyłby, że sytuacje na prawej i dolnej bandzie są ze sobą powiązane. Po ruchu Białych ③ połączyłby się w *a*, a po ruchu Białych ⑦ połączyłby się

w *b*. W rezultacie Czarne skończyłyby z terytorium mniejszym o jeden punkt. Co więcej, Białe *wyszłyby w sente* (zachowałyby inicjatywę i mogłyby zagrać w innej części planszy).

Żeby poprawnie analizować przedstawiony przykład, sytuacje na prawej i dolnej bandzie trzeba potraktować jako jedną większą lokalną sytuację. Tego typu zależności między sytuacjami w prawdziwych partiach występują bardzo często. Zauważenie, w jaki sposób lokalne sytuacje się ze sobą łączą, często wymaga dużej wiedzy eksperckiej. Dlatego nawet po opracowaniu precyzyjnej teorii końcówek stworzenie algorytmu do wybierania poprawnych zagrań w fazie końcowej jest nadal bardzo trudnym zadaniem.

Omówienie

Wartości ruchów obliczone zgodnie z klasyczną metodą stanowią dużą pomoc przy wyborze ruchów w końcówce. Jednakże czasami najlepszy ruch nie jest tym o najwyższej wartości. Optymalna gra w fazie końcowej jest trudna do zalgorytmizowania. David Wolfe [18] dowiódł, że w ogólnym przypadku jest to zagadnienie PSPACE-trudne.

Jak zobaczymy w podrozdziale 2.5, kombinatoryczna teoria gier wypracowała narzędzia pozwalające na bardziej precyzyjną analizę sytuacji niż samo liczenie wartości ruchów. Przy użyciu nowej teorii dla mniej skomplikowanych pozycji znaleziono względnie prosty algorytm podejmowania poprawnych decyzji.

Żeby móc efektywnie stosować narzędzia matematyczne do pozycji występującej w fazie końcowej, gracz potrzebuje wiedzy eksperckiej pozwalającej na znalezienie dobrych zagrań w lokalnych sytuacjach. W przypadku programów grających wiedzę tę teoretycznie można by zastąpić dokładnym przeszukiwaniem drzewa gry dla każdej lokalnej sytuacji. Jednakże dużą trudność dla utworzenia takiego programu będzie stanowić kwestia samego podzielenia planszy na odizolowane sytuacje. Jak zobaczyliśmy na diagramie 2.11, sytuacje wyglądające na odizolowane, mogą być w rzeczywistości wzajemnie powiązane. Odnajdowanie takich powiązań jest bardzo trudne do zalgorytmizowania.

2.5. Zastosowanie kombinatorycznej teorii gier

Kombinatoryczna teoria gier jest to gałąź matematyki zajmująca się formalnym opisem określonej klasy gier. Są to gry o doskonałej informacji bez czynnika losowego, w których dwaj gracze naprzemiennie wykonują ruchy. Poza tym gry te są skończone, a zwycięzcą zostaje gracz, który wykona ostatni ruch.

W Go celem gry jest zajęcie większej części planszy niż przeciwnik. Okazuje się jednak, że zasady można minimalnie zmodyfikować tak, aby celem gry stało się wykonanie ostatniego ruchu, a wynik nie uległ zmianie. W zmodyfikowanych regułach nie można pasować. Od chwili, w której przy zastosowaniu zwykłych reguł gracze by spasowali, partia wchodzi w fazę wypełniania terytoriów. Od tej pory legalny ruch gracza to albo położenie kamienia wewnątrz swojego terytorium, (nawet jeśli w zwykłych zasadach byłby to ruch samobójczy), albo oddanie przeciwnikowi jednego ze zbitych kamieni. Uważny Czytelnik sprawdzi, że zasady te wyłaniają tego samego zwycięzcę, co zwykłe reguły Go.

Kombinatoryczna teoria gier określa grę jako abstrakcyjny obiekt matematyczny. Kiedy pozycja na planszy do Go składa się z odizolowanych sytuacji, każdą z nich można opisać jako osobną grę. Globalna sytuacja jest wówczas grą stanowiącą ich sumę. Takie ujęcie bardzo upraszcza analizę pozycji i pozwala na odnalezienie optymalnych zagrań. Elwyn Berlekamp i David Wolfe w książce *Mathematical Go: Chilling Gets the Last Point* [4] dokładnie opracowali teorię rozgrywania jednopunktowych końcówek w oparciu o kombinatoryczną teorię gier.

Definicja gry według kombinatorycznej teorii gier

Definicja 2.5.1. *Gra G jest to para uporządkowana zbiorów gier* $\{\mathbb{G}_L, \mathbb{G}_R\}$.

Wygodną reprezentację gry stanowi drzewo, w którym każdy węzeł ma dzieci lewe ($\mathbb{G}_L$) oraz dzieci prawe ($\mathbb{G}_R$). Gra może być rozgrywana przez dwóch graczy: Lewego i Prawego, którzy naprzemiennie wykonują ruchy. Pierwszy ruch w partii może być wykonany albo przez gracza Lewego, albo Prawego. Gracza, który wykonuje pierwszy ruch, nazywa się pierwszym, a drugiego drugim.

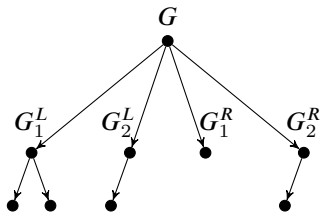


Diagram 2.12: Drzewo gry $G = \{*, 1\}|\{0, 1\}$

Używając reprezentacji drzewa, ruch należy rozumieć jako przejście z węzła do jednego z jego dzieci. Lewy gracz może przejść do jednego z lewych dzieci, Prawy - do jednego z prawych. Jeśli w pewnym momencie jeden z graczy nie ma dostępnego żadnego ruchu, przegrywa partię. Partia gry jest to więc przejście od korzenia do jednego z liści drzewa.

Często używana skrótowa notacja przedstawia korzeń drzewa jako pionową kreskę:

$$G = \mathbb{G}^L|\mathbb{G}^R = \{G_1^L, G_2^L, \dots\}|\{G_1^R, G_2^R, \dots\}$$

W kombinatorycznej teorii gier liczby definiuje się jako specjalny rodzaj gier. Jeśli oba zbiory gier $\mathbb{G}^L$ oraz $\mathbb{G}^R$ są puste, to piszemy $G = \emptyset|\emptyset = 0$. Piszemy również $G = 0|\emptyset = 1$ i ogólnie $G = 0|n-1) = n$. Nie wszystkie gry to liczby. Na przykład grę $G = 0|0$ nazywamy gwiazdką (ang. *star*) i piszemy $G = *$.

Żeby przedstawić w tekście głębsze drzewo, węzły bliższe korzeniowi reprezentuje się przy pomocy większej liczby pionowych kresek. Na przykład dla drzewa z diagramu 2.12 piszemy:

$$G = \{*, 1\}|\{0, 1\} = \{0|0, 0|\emptyset\}||\{\emptyset|\emptyset, 0|\emptyset\}$$

Gra przeciwna, suma gier

Dla gry G definiuje się grę przeciwną $-G$. Drzewo gry $-G$ to odbicie lustrzane drzewa gry G. Jest to więc gra, w której gracze Lewy i Prawy zamienili się rolami. Formalnie,

$$-G = -\mathbb{G}^R | -\mathbb{G}^L, \text{ gdzie } -\mathbb{G} = -G_{G\in\mathbb{G}}$$

Na przykład $-0 = 0$ oraz $-* = -\{0|0\} = \{-0| - 0\} = \{0|0\} = *$. Dla gry będącej liczbą naturalną $G = n$ piszemy $-G = -n = -(n-1)|0$.

Dla dwóch gier G, H definiujemy grę $G+H$ będącą ich sumą. O partii gry $G+H$ najłatwiej myśleć jako o jednoczesnym rozgrywaniu przez graczy partii G i partii H. W każdym ruchu gracz może wykonać ruch w jednej z tych gier i pozostawić drugą nietkniętą. Formalnie sumę gier definiuje się jako:

$$G + H = \left\{(G^L + H) \cup (G + H^L)\right\}_{G^L\in\mathbb{G}^L, H^L\in\mathbb{H}^L} | \left\{(G^R + H) \cup (G + H^R)\right\}_{G^R\in\mathbb{G}^R, H^R\in\mathbb{H}^R}$$

Sumę gier skrótowo zapisuje się bez znaku plusa $GH = G + H$. Różnica gier G i H to suma gier G i $-H$. Piszemy $G - H = G + (-H)$

Gry występujące w fazie końcowej

W fazie końcowej Go każdą lokalną sytuację może potraktować jako osobną grę. Wówczas całoplanszową pozycję traktujemy jako sumę tych gier. Przyjmujemy, że Czarne są graczem Lewym, a Białe prawym. Przyjmujemy również, że gramy w Go z dodatkową fazą wypełniania terytoriów. Wówczas jednopunktowe terytorium Czarnych jest grą $G = 0|\emptyset = 1$, ponieważ zapewnia ono jeden ruch dostępny Czarnym i żadnych ruchów dostępnych Białym. W ogólności terytorium Czarnych jest to liczba całkowita dodatnia, a terytorium Białych to liczba całkowita ujemna. Punkt neutralny jest to gwiazdka $* = 0|0$, bo mogą tam zagrać zarówno Czarne, jak i Białe, i będzie to ostatni ruch w lokalnej sytuacji.

Diagram 2.13 przedstawia przykłady nieskomplikowanych pozycji w końcówce. Każdy z ruchów Czarnych a, b, c jest wart 1 punkt i z punktu widzenia teorii przedstawionej w poprzednich podrozdziałach są one równoważne. Jeśli jednak chcielibyśmy zbudować drzewa gry dla przedstawionych sytuacji, to musielibyśmy uwzględnić wszystkie możliwe ruchy i uzyskane drzewa byłyby bardzo skomplikowane. Tylko sytuacja na środku miałaby nieduże drzewo postaci $2|0$. Chcielibyśmy, żeby teoria pozwoliła nam zapomnieć o nieistotnych różnicach między tymi sytuacjami. W tym celu kombinatoryczna teoria gier wprowadza metodę upraszczania postaci gier oraz operację studzenia.

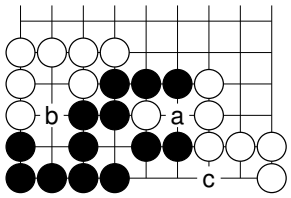


Diagram 2.13: Ruchy warte 1 punkt

Upraszczanie postaci gier

Kombinatoryczna teoria gier wprowadza częściowy porządek na grach. Dla gier G, H, zależnie od tego, kto przy optymalnej grze wygrywa w grze $G - H$, piszemy:

$G > H$, gdy w grze $G - H$ wygrywa gracz lewy
$G = H$, gdy w grze $G - H$ wygrywa gracz drugi
$G < H$, gdy w grze $G - H$ wygrywa gracz prawy
$G \gtrless H$, gdy w grze $G - H$ wygrywa gracz pierwszy

Na przykład $0 < 1$, bo w grze $0 - 1 = -1$ wygrywa gracz prawy. Przykład pary nieporównywalnych gier to * i 0: $* \gtrless 0$, bo w grze $* - 0 = * = 0|0$ wygrywa gracz pierwszy.

Dzięki wprowadzeniu relacji częściowego porządku można zdefiniować dwie operacje upraszczające pełne drzewo gry.

Pierwsza z operacji to usuwanie zdominowanych opcji. Oznacza ono usunięcie z rozważań tych zagrań, które prowadzą do nie lepszego rezultatu niż pewne inne zagranie. Na przykład na diagramie 2.13 po lewej stronie każdy ruch inny niż b daje gorszy wynik. Dlatego jedyną opcją, którą zachowamy w drzewie gry zarówno dla Czarnych, jak i dla Białych, będzie zagranie w b.

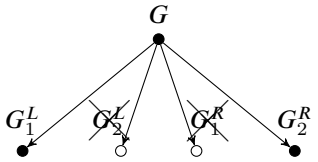


Diagram 2.14: Usuwanie zdominowanych opcji, $G_1^L \geq G_2^L$, $G_2^R \leq G_1^R$

Formalnie, ze zbioru lewych opcji $\mathbb{G}^L$ usuwamy wszystkie takie gry G^L, dla których istnieje gra $G'^L \in \mathbb{G}^L$ taka, że $G'^L \geq G^L$. Natomiast ze zbioru $\mathbb{G}^R$ usuwamy takie G^R, dla których istnieje takie $G'^R \in \mathbb{G}^R$, że $G'^R \leq G^R$. Użyta jest nierówność w drugą stronę, ponieważ gry mniejsze są lepsze dla Białych.

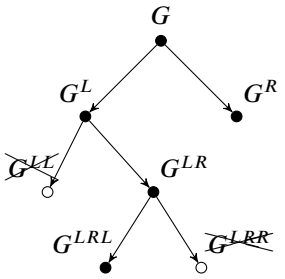


Diagram 2.15: Skracanie gałęzi, $G^{LR} \leq G$

Druga operacja to skracanie gałęzi (ang. *reversing reversible options*). Uważny czytelnik zwróci uwagę na podobieństwo między tą operacją a definicją przywileju przytoczoną w podrozdziale 2.2. Skracania gałęzi dokonuje się, jeśli po ruchu jednego gracza odpowiedź drugiego prowadzi do sytuacji nie lepszej niż wyjściowa. Wówczas opcje pierwszego gracza po tej wymianie zostają uznane za dzieci korzenia drzewa, a opcje drugiego gracza zostają usunięte z drzewa.

Przykładowo, na dolnej bandzie po ruchu Czarnych w c i atari Białych sytuacja jest nie lepsza dla Czarnych niż na początku. Skracając gałęzie drzewa, jako opcję Czarnych należy potraktować całą sekwencję trzech ruchów (c - atari - połączenie). Natomiast ruchy Czarnych inne niż c zostaną usunięte jako zdominowane opcje.

Uproszczenie gry G przy pomocy usuwania zdominowanych opcji i skracania gałęzi sprowadza ją do postaci kanonicznej G'. W częściowym porządku na grach zachodzi równość $G = G'$. Gra G' jest wyznaczona jednoznacznie [4].

Dla gier przedstawionych na diagramie 2.13 związanych z przecięciami a, b i c postaci kanoniczne to odpowiednio $2|0$, $1* \,|\, -1*$ oraz $1|-1$.

Studzenie gier

Uzyskane postaci kanoniczne gier z diagramu 2.13 nadal się od siebie różnią. Różnice pomiędzy tymi postaciami dotyczą jednak wyłącznie pól neutralnych (czyli gwiazdek) oraz punktów terytorium (czyli liczb całkowitych). W partii Go w momencie, gdy na planszy pozostały wyłącznie terytoria i punkty neutralne, rozgrywka staje się trywialna. Żeby skupić uwagę na elementach fazy końcowej, które nie są trywialne, Berlekamp i Wolfe [4] stosują do Go operację z kombinatorycznej teorii gier nazywaną studzeniem.

Gra ostudzona[3] o pewną liczbę rzeczywistą t jest to gra G_t, w której na wykonanie ruchu nałożona jest opłata w wysokości t, podobnie jak w ekonomicznej wersji zasad:

$$G_t = G_t^L - t | G_t^R + t$$

W książce *Mathematical Go* [4] autorzy analizują sytuacje, w których ruchy mają wartość co najwyżej jednego punktu. Analizę tę ułatwia ostudzenie sytuacji o 1 punkt.

W ostudzonej grze Go graczom nie opłaca się wypełniać neutralnych pól. Neutralne pole (czyli gra $G = *$) ostudzone o 1 jest równe $G_1 = 0$. Sytuacja po lewej stronie na diagramie 2.13 po ostudzeniu zapisuje się jako

$$G_1 = \{1 * | -1*\}_1 = 0|0 = *$$

Tak samo jest w przypadku sytuacji na dolnej bandzie:

$$G_1 = \{1| -1\}_1 = 0|0 = *$$

Do opisu sytuacji na środku planszy wykorzystamy stwierdzenie 2.1.3. Gdybyśmy liczyli lokalny wynik w tej sytuacji zwyczajnym sposobem, otrzymalibyśmy

$$L = \frac{1}{2}(L_C + L_B) = \frac{1}{2}(2 + 0) = 1$$

Analiza uprości się jednak, jeśli od każdego z rozważanych wyników odejmiemy 1. Powiemy, że po ruchu Czarnych lokalny wynik jest równy 1, a po ruchu Białych jest równy -1. Obecny lokalny wynik jest więc równy $L = 0$. Drzewo gry związanej z tą sytuacją ma więc postać $G = 1| - 1$, a po ostudzeniu zachodzi

$$G_1 = \{1| -1\}_1 = 0|0 = *$$

Po sprowadzeniu gier z diagramu 2.13 do postaci kanonicznej i ostudzeniu, dla każdej gry otrzymujemy taką samą postać. Wprowadzone narzędzia pozwoliły zapomnieć o nieistotnych różnicach pomiędzy sytuacjami.

[3] Formalna definicja znajduje się w [3], ss. 147 - 149

Jednopunktowe końcówki

W niniejszym podrozdziale będziemy rozważali sytuacje sprowadzone do postaci kanonicznej i ostudzone. Zamiast G_1 będziemy po prostu pisali G. Jak zobaczymy, nie wszystkie gry, w których ruch ma wartość 1 punkt, mają taką samą postać. Oprócz wspomnianych w poprzednim podrozdziale gwiazdek, istnieją ruchy, które są sente dla jednego z graczy. Jeszcze inne ruchy stanowią przypadek graniczny w definicji sente, to znaczy pewny zysk jest równy wartości kontynuacji.

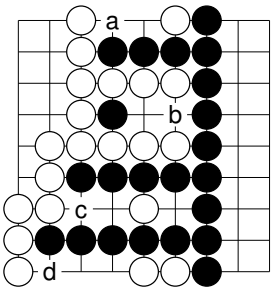


Diagram 2.16: Ruchy nieskończenie małe (po ostudzeniu)

Przykład takiego granicznego przypadku stanowi sytuacja na górnej bandzie na diagramie 2.16. Dla uproszczenia analizy od wszystkich lokalnych wyników będziemy odejmowali 2, dzięki czemu obecny lokalny wynik będzie równy 0. Po ruchu Czarnych w a wynik Czarnych poprawi się o 1 punkt, ale Czarne będą musiały uiścić opłatę w wysokości 1. Po ruchu Białych wynik Czarnych będzie równy -1, ale to Białe będą musiały uiścić opłatę. Dlatego gra ma postać kanoniczną $G = 0||0|0 = 0|*$. Gra $0|*$ nazywa się wyżem (ang. *up*). Piszemy $G = 0|* = \uparrow$.

W okolicy przecięcia b widzimy grę przeciwną do wyżu (uzyskaną poprzez zamianę kolorów). Taka gra nazywa się niżem (ang. *down*) i piszemy $G = *|0 = \downarrow$.

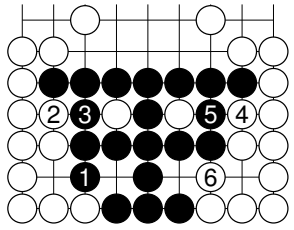


Diagram 2.17: Wyż i gwiazdka - gry nieporównywalne

Gwiazdka, wyż i niż stanowią wartości nieskończenie małe, to znaczy większe od wszystkich liczb ujemnych i mniejsze od wszystkich liczb dodatnich. O ile jednak gwiazdka jest nieporównywalna z zerem, o tyle wyż jest dodatni, a niż ujemny.

Wyż i gwiazdka są nieporównywalne. Aby to udowodnić, należy dwa razy rozegrać partię gry $\uparrow -*$. Ilustruje to diagram 2.17, na którym znajdują się dwie kopie gry $\uparrow -*$, jedna po lewej, a druga po prawej stronie planszy. Na pierwszą partię składają się ruchy ❶-②-❸, a na drugą ruchy ④-❺-⑥. W pierwszej partii zaczynają Czarne, a w drugiej Białe. W obu przypadkach gracz, do którego należy pierwszy ruch, wykonuje również ostatni ruch w partii, czyli wygrywa. W grze $\uparrow -*$ zwycięża pierwszy gracz, zatem zgodnie z określeniem częściowego porządku na grach zachodzi $\uparrow \gtrless *$.

Suma dwóch wyżów to dwójwyż (ang. *double-up*). Jest on oznaczany symbolem $\Uparrow = \uparrow + \uparrow$. Okazuje się, że dwójwyż jest większy niż gwiazdka. Dowód tego faktu przedstawia diagram 2.18. Na diagramie rozegrane są dwie partie gry $\Uparrow -*$. Na pierwszą składają się ruchy od ❶ do ❺, a na drugą ruchy od ⑥ do ❾ W obu partiach Czarne wykonują ostatni ruch, co dowodzi, że $\Uparrow > *$.

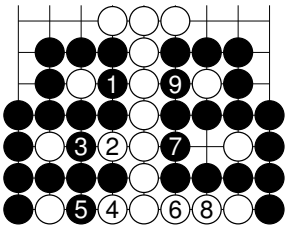


Diagram 2.18: Dwójwyż przewyższa gwiazdkę

Ruch c na diagramie 2.16 stanowi sente dla Białych. Od wszystkich wyników w tej sytuacji odejmujemy 3, dzięki czemu obecny lokalny wynik równa się 0. Jeżeli Białe zagrają dwa ruchy z rzędu, to łącznie uiszczą opłaty w wysokości 2, ale zabiorą Czarnym 3 punkty i wynik będzie równy 1 punkt na korzyść Białych. Dlatego gra ma postać $G = 0||0| - 1$. Taka gra jest nazywana plusikiem (ang. *tiny*), a dokładniej plusową jedynką (ang. *tiny one*). Piszemy $G = 0||0| - 1 = +_1$. Ogólnie, $+_r = 0||0| - r$ dla dowolnej liczby $r > 0$. Przeciwieństwem plusika jest minusik (ang. *miny*): $-_r = r|0||0$ i można Go uzyskać, zamieniając kolory w sytuacji będącej plusikiem.

Każdy plusik jest dodatni, jednak mniejszy od wyżu. Intuicyjnie można ten fakt rozumieć w taki sposób, że Czarne mają mniejszą szansę na zagranie ruchu c niż a. Zachodzi również $+_r < +_s$ dla $0 < r < s$.

Korytarz na dolnej bandzie na diagramie 2.13 to gra postaci $G = 0|+_2$. Dla dłuższych korytarzy używa się skróconego zapisu $0^n|+ = 0||\ldots|0\ldots0|+$.

Autorzy książki *Mathematical Go* przeanalizowali gry opisanych typów (gwiazdki, wyże, plusiki itp.) i ustalili na nich częściowy porządek. Owocem tej pracy jest algorytm poprawnego grania jednopunktowych końcówek. Czarne w pierwszej kolejności powinny wybrać ruchy w minusikach (są to ruchy sente, dla Czarnych mają więc wartość przekraczającą jeden punkt). Następnie powinny atakować korytarze zakończone minusikami, takie jak na dolnej bandzie na diagramie 2.16. W dalszej kolejności powinny grać na niżach bądź korytarzach zakończonych niżami (kolejność nie ma znaczenia). Po zagraniu wszystkich tych ruchów trzeba podjąć decyzję, czy blokować własne korytarze, czy też grać w gwiazdce. Jeśli liczba gwiazdek jest nieparzysta, poprawny jest ruch w gwiazdce, a w przeciwnym razie blokada korytarza. Ruchy o wartościach niższych niż 1 punkt należy grać na samym końcu.

Większość z ustaleń Berlekemapa i Wolfe'a jest zgodna z intuicjami silnych graczy Go. Jeden wynik jest jednak zaskakujący. Dotyczy on kolejności wchodzenia w korytarze. Dla dwóch korytarzy zakończonych minusikami zacząć należy od wchodzenia w korytarz, na którego końcu jest mniej do zyskania.

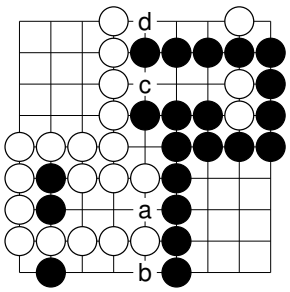


Diagram 2.19: Gra $G - G$
Najlepszy ruch dla Czarnych to b, najlepszy ruch dla Białych to d

Dowód przebiega przez rozważenie gry złożonej z dwóch takich korytarzy $G = \{-_r|0^m\} + \{-_s|0^n\}$. Aby sprawdzić, która strategia wchodzenia w korytarze jest bardziej opłacalna należy rozważyć dwie partie gry $G - G$. Jeden z graczy będzie wchodził najpierw w korytarze zakończone mniejszym zyskiem, a drugi - większym. Okazuje się, że zależnie od tego, kto gra pierwszy ruch, partia kończy się albo remisem, albo porażką gracza wchodzącego w korytarz zakończony bardziej wartościowym ruchem. Wynik ten wydaje się wysoce nieintuicyjny nawet silnym graczom Go.

Niepołączone łańcuchy z dostępem do wielu korytarzy

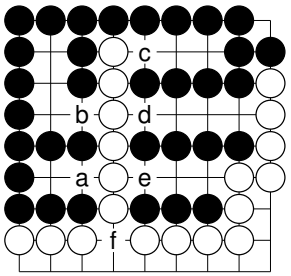


Diagram 2.20: Dostęp do wielu korytarzy

Przedstawiliśmy opisy wielu często widywanych sytuacji, w których wartość ruchu nie przekracza jednego punktu. Kombinatoryczna teoria gier posłużyła Berlekampowi i Wolfe'owi do opisu jeszcze jednej klasy takich sytuacji. Są to niepołączone łańcuchy z dostępem do wielu korytarzy. Na diagramie 2.20 łańcuch białych kamieni na środku planszy nie jest połączony z pozostałymi dwoma łańcuchami Białych. Kiedy zostaną mu zabrane wszystkie oddechy zewnętrzne, Białe będą zmuszone połączyć się w f. Jednocześnie Białe mają możliwość wydłużania swojego łańcucha przez wchodzenie w kilka korytarzy, z których jedne są zablokowane na drugim końcu przez czarny kamień, a inne nie. Obliczenie lokalnych wyników w tych korytarzach bez pomocy narzędzi teoriogrowych stanowi olbrzymie wyzwanie.

Berlekamp i Wolfe podali algorytm liczenia lokalnych wyników w takich sytuacjach.

Pierwszy krok polega na obliczeniu lokalnych wyników w korytarzach bez uwzględnienia faktu, że łańcuch Białych jest niepołączony. Tak obliczone wyniki dla zablokowanych korytarzy oznaczymy przez b z indeksami dolnymi (na diagramie 2.20 będą to b_a, b_b i b_c). Wyniki dla niezablokowanych korytarzy oznaczymy literą u (na diagramie będą to u_d oraz u_e). Piszemy $B = \Sigma b$.

Jeżeli $B \geq 1$, to b oraz u są poprawnymi lokalnymi wynikami w każdym z korytarzy. W przeciwnym przypadku trzeba będzie je trochę zmodyfikować.

Jeżeli $B < 1$ oraz nie ma żadnych niezablokowanych korytarzy, to lokalny wynik w sytuacji będącej sumą wszystkich korytarzy ma taką samą postać kanoniczną jak pewien pojedynczy korytarz zakończony minusikiem.

Jeśli wreszcie $B < 1$ oraz istnieją niezablokowane korytarze, to wynik $s = \min u$ w najkrótszym z nich należy podzielić przez 2. Wyniki w pozostałych niezablokowanych korytarzach pozostają bez zmian. Z kolei wynik w każdym z zablokowanych korytarzy należy przemnożyć przez $\frac{s}{2}$.

Policzenie lokalnych wyników w korytarzach, do których dostęp ma niepołączony łańcuch, pozwala na obliczenie wartości ruchów w takiej sytuacji. Berlekamp i Wolfe skonstruowali całoplanszowy problem[4], w którym występuje kilka tego typu sytuacji oraz kilka innych, w których wartość ruchu nie przekracza jednego punktu. Problem ten zademonstrował siłę narzędzi wypracowanych przez kombinatoryczną teorię gier. Został on pokazany kilku czołowym graczom na świecie, którzy spędzili nad nim do kilku godzin. Mimo długich rozważań żaden z profesjonalistów nie odnalazł poprawnego rozwiązania. Profesjonaliści grali przeciwko matematykom partie zaczynające się od tej pozycji. Każda partia, niezależnie od tego, kto grał którym kolorem, zakończyła się zwycięstwem matematyków kierujących się nieskomplikowanym algorytmem. Pozycja ta została nazwana problemem peszącym zawodowców.

[4]Diagram przedstawiający problem znajduje się w podrozdziale 4.3.

Podsumowanie

Faza końcowa w Go jest tą częścią gry, która w największym stopniu poddaje się opisowi matematycznemu. Gracze Go wypracowali nieskomplikowaną arytmetyczną metodę obliczania wartości ruchów w końcówce. Obliczone wartości stanowią dużą pomoc przy wyborze zagrań. Jednakże nie zawsze ruch o największej wartości jest tym poprawnym. W ogólnym przypadku zagnienie końcówek goistycznych nie daje się łatwo zalgorytmizować (jest PSPACE-trudne).

Goistyczne końcówki były jedną z inspiracji dla stworzenia kombinatorycznej teorii gier w drugiej połowie XX wieku. Teoria wypracowała silne narzędzia, które mają zastosowanie do szerokiej klasy gier. W przypadku gry w Go okazało się, że klasyczna metoda liczenia wartości ruchów nie wystarcza do precyzyjnej analizy pozycji występujących w końcówce. Kombinatoryczna teoria gier pozwoliła na czynienie subtelnych rozróżnień pomiędzy zagraniami o tej samej wartości. Berlekamp i Wolfe na bazie tej teorii wypracowali algorytm liczenia wartości ruchów w sytuacji łańcuchów posiadających dostęp do wielu korytarzy. Dla takich ruchów liczenie wartości klasyczną metodą staje się niesłychanie skomplikowane. Silni gracze nie posiadają też dobrych intuicji odnośnie takich sytuacji.

Mimo wszystko, kombinatoryczna teoria gier ma ograniczone zastosowanie w Go. Zalgorytmizowanie podejmowania poprawnych decyzji w fazie końcowej nie jest łatwe. W przypadku ruchów o większych wartościach niż jeden punkt obliczenia zalecane przez teorię stają się bardziej skomplikowane. Inną przeszkodą w stosowaniu teorii jest występowanie w Go walk ko. Sytuacje te lokalnie nie stanowią gier skończonych w przeciwieństwie do gier, którymi zajmuje się teoria. Matematycy w późniejszych pracach podjęli próbę rozszerzenia kombinatorycznej teorii gier również na takie przypadki [2, 15]. Jedną z największych trudności stanowi jednak samo podzielenie planszy na odizolowane lokalne sytuacje. Do ustalenia, czy sytuacje są ze sobą wzajemnie powiązane, konieczna jest goistyczna wiedza ekspercka.

Niektórzy badacze zajmujący się kombinatoryczną teorią gier upatrywali w niej nadziei dla komputerowego Go [4, 11]. Jednak w rzeczywistości postęp w komputerowym Go dokonał się w inny sposób. Następny rozdział opisuje rozwój narzędzi programistycznych, który ostatecznie doprowadził do stworzenia programów pokonujących najsilniejszych graczy na świecie.

Rozdział 3

Programy grające oparte o głębokie sieci konwolucyjne

Klasyczne programy grające w gry dokonują wyboru ruchu na podstawie przewidywań dotyczących dalszego rozwoju partii. Program buduje drzewo gry i dokonuje wyceny pozycji końcowych. Dysponując tymi wycenami, można ustalić, które z rozważanych wariantów składają się z optymalnych zagrań, i podjąć decyzję, który ruch w obecnej sytuacji jest najlepszy. Klasyczny algorytm służący do podejmowania tej decyzji nazywa się minimax. Odcinanie alfa-beta jest usprawnieniem algorytmu minimax, które pozwala na odnalezienie najlepszego ruchu bez budowania pełnego drzewa gry. Jeśli po pewnym ruchu wynik zawsze jest lepszy niż po innym ruchu i pewnej odpowiedzi przeciwnika, to tego drugiego ruchu w ogóle nie trzeba rozważać i tworzyć od niego nowych gałęzi. Dla mało skomplikowanych gier, np. kółko i krzyżyk, algorytm szybko odnajduje optymalne ruchy.

W przypadku skomplikowanych gier zastosowanie samego minimaxu z odcinaniem alfa-beta jest niewystarczające. Algorytmu nie da się wykonać w czasie rzeczywistym nawet przy zastosowaniu dużej mocy obliczeniowej. W takich sytuacjach z pomocą może przyjść heurystyka oceniająca pozycję na planszy. Program nie buduje gałęzi drzewa gry aż do stanu końcowego. Zamiast tego tworzy na przykład gałęzie o długości 3, to znaczy rozważa wszystkie trzyruchowe sekwencje w danej sytuacji. Jeśli program dysponuje heurystyką, która trafnie oceni pozycje po tych trzech ruchach, takie przeszukiwanie może dostarczyć dużo informacji i prowadzić do wyboru dobrych zagrań (choć już niekoniecznie najlepszych).

W przypadku szachów, mimo wielkiego poziomu skomplikowania gry, możliwe okazało się utworzenie trafnych heurystyk, bazujących między innymi na liczbie bierek każdego koloru pozostałych na planszy oraz ich względnych wartości. Algorytm alfa-beta wzbogacony o heurystyki był rozwiązaniem użytym w programie Deep Blue. Program ten w 1996 roku jako pierwszy w historii pokonał szachowego mistrza świata. Oprócz wspomnianego algorytmu przeszukiwania Deep Blue wykorzystywał obszerną wiedzę dziedzinową oraz bazę otwarć.

Stworzenie silnego programu grającego w Go okazało się jednak z kilku powodów trudne. Po pierwsze, dużą przeszkodą dla algorytmu przeszukującego jest szerokość drzewa gry. Przy każdym ruchu w Go kamień można postawić na dowolnym niezajętym przecięciu, z pominięciem co najwyżej kilku zagrań (ruchów samobójczych i odbicia ko). W związku z tym nawet na małej planszy (9×9) liczba dostępnych opcji jest większa niż na przykład w szachach.

Po drugie, dla Go trudne okazało się utworzenie heurystyki oceniającej pozycję na planszy. Konsultacja z silnymi graczami niewiele pomaga, ponieważ swoją ocenę opierają oni głównie na intuicji. Tłumaczenie tych intuicji możliwe jest przy użyciu rozmaitych pojęć goistycznych. Jednakże pojęcia te odnoszą się do wysokopoziomowego opisu pozycji i trudno przełożyć je na

język zrozumiały dla komputera. Jedynym elementem Go, który doczekał się rozbudowanego opisu formalnego, jest teoria końcówek omówiona w rozdziale 2. Jest to jednak tylko jeden, bynajmniej nie najważniejszy aspekt gry.

Podobnie jak w przypadku szachów, również w Go program może skorzystać na posiadaniu bazy standardowych rozegrań występujących w otwarciu. Jednak w przypadku Go rozegrania takie, zwane *joseki*, w większości ograniczają się do sytuacji w narożniku. Żeby podejmować dobre decyzje w otwarciu, znajomość joseki jest niewystarczająca. Wybór joseki należy bowiem uzależnić od całoplanszowej pozycji. Ponadto, nie da się dokonać pełnej klasyfikacji joseki. Cały czas bowiem najsilniejsi gracze analizują nowe warianty, obalając stare joseki i wymyślając nowe.

Do programów często oprócz bazy joseki dołączano również bazę *kształtów*. Próbowano klasyfikować lokalne konfiguracje, które mogą wystąpić gdziekolwiek na planszy (np. na obszarze wymiarów 3×3) i oceniać je pod względem efektywności ułożenia kamieni.

W komputerowym Go jednym z najbardziej znanych programów opartych na klasycznych rozwiązaniach było Many Faces of Go. Poza algorytmem alfa-beta i wiedzą dziedzinową służącą do oceniania pozycji posiadał on bazę kształtów oraz bazę joseki [9]. Many Faces of Go w latach dziewięćdziesiątych był jednym z najsilniejszych programów. Jednakże był to poziom niewystarczający nawet do pokonania średniozaawansowanych amatorów.

Kilka nowatorskich rozwiązań doprowadziło do kolejnych przełomów w komputerowym Go. W podrozdziale 3.1 przedstawiam metodę Monte Carlo, która posłużyła jako niekonwencjonalna heurystyka dla pozycji występujących w grze. Omawiam również algorytm Monte Carlo Tree Search, który był sprytną metodą przeszukiwania drzewa gry, dobrze radzącą sobie nawet w przypadku gry o tak szerokim drzewie jak Go.

W podrozdziale 3.2 opisuję zastosowanie sieci neuronowych w komputerowym Go, które zapewniło programom dobrą znajomość wzorców występujących w grze (kształtów oraz joseki) bez konieczności tworzenia ich baz. Pierwszym programem grającym na poziomie profesjonalnym stało się AlphaGo, w którym wykorzystano sieci neuronowe w połączeniu z Monte Carlo Tree Search.

W podrozdziale 3.3 omawiam elegancki i mało skomplikowany algorytm zaprezentowany przez twórców AlphaGo, który udoskonalił poprzednio użyte rozwiązania i doczekał się wielu reimplementacji. Został on nazwany AlphaGo Zero.

W podrozdziale 3.4 prezentuję nowy otwartoźródłowy projekt KataGo, który ma kilka przewag nad wcześniejszymi programami. W szczególności KataGo zapewnia bardziej rozbudowany interfejs użytkownika, dzięki któremu możliwe jest przeprowadzenie eksperymentów opisanych w rozdziale 4.

3.1. Od Monte Carlo do Monte Carlo Tree Search

Monte Carlo jest to metoda modelowania złożonego procesu poprzez przeprowadzenie wielu losowych symulacji. W przypadku gier jest to rozgrywanie partii złożonych z losowych zagrań. Metoda Monte Carlo może posłużyć do dokonania oceny stanu gry poprzez uśrednienie wyników rozegranych losowych partii. Stając przed wyborem następnego ruchu, program może rozważyć wszystkie możliwe zagrania i wycenić pozycję po każdym z nich przy pomocy opisanego sposobu. Następnie program wykona ruch, po którym pozycja została oceniona jako najbardziej korzystna.

W 1993 roku Bernd Brügmann opublikował pionierską pracę [6] dotyczącą zastosowania metody Monte Carlo w komputerowym Go. Żeby losowe rozgrywki mogły dobiec końca, w swoim programie Gobble Brügmann musiał zaimplementować pewną minimalną wiedzę

dotyczącą mechaniki gry. Gobble prowadząc symulacje, nigdy nie wybierał pasu. Miał jednak zabronione wykonywać ruch zapełniający jednopunktowe oko. W momencie, w którym zostałyby tylko takie ruchy, program pasował.

W symulowanych rozgrywkach ruchy nie były losowane z równym prawdopodobieństwem. Twórca programu wysunął hipotezę, że dla ustalonej pozycji na planszy pewne pola są kluczowe. W związku z tym zagranie na takim polu będzie dobrym posunięciem niezależenie, w którym momencie zostanie wykonane. Program Gobble prowadził więc dla każdego dostępnego ruchu statystykę, jakim wynikiem kończyły się partie zawierające ten ruch. Zagrania w symulowanych rozgrywkach były losowane zgodnie z tymi statystykami.

Gobble grał na poziomie gracza początkującego. Czyniło go to o kilka poziomów słabszym od Many Faces of Go. W następnych latach programiści eksperymentowali z różnymi modyfikacjami algorytmu Brügmanna [5]. Jednym z pomysłów było dodanie do programu bazy kształtów, która pozwalałaby na wybór bardziej rozsądnych zagrań w losowych rozgrywkach. Sposób wyboru losowych zagrań musiał jednak pozostać stosunkowo mało skomplikowany, żeby było możliwe przeprowadzenie wielu rozgrywek w czasie rzeczywistym.

Metoda Monte Carlo zyskała na znaczeniu po tym, jak w 2006 Levente Kocsis i Csaba Szepesvári zaprezentowali nowy sposób przeszukiwania drzewa gry, który został nazwany Monte Carlo Tree Search (w skrócie MCTS). Algorytm MCTS buduje drzewo możliwych wariantów dla danej sytuacji na planszy w sposób iteracyjny. Na jedną iterację składają się cztery kroki: selekcja, ekspansja, ewaluacja i wsteczna propagacja wyników.

1. **Selekcja** stanowi przejście od korzenia drzewa (reprezentującego obecną sytuację na planszy) do jednego z liści (reprezentującego sytuację po kilku zagranych ruchach). W każdym z kolejno odwiedzanych węzłów decyzja o tym, do którego dziecka przejść, podejmowana jest na podstawie statystyk przechowywanych w każdym z dzieci zgodnie z formułą matematyczną UCB.

2. **Ekspansja** odbywa się dla wybranego w poprzednim kroku liścia. Polega na dodaniu do drzewa pewnej ustalonej liczby dzieci tego liścia (odpowiadających różnym ruchom dostępnym w sytuacji reprezentowanej przez ten liść).

3. **Ewaluacja** polega na wycenie sytuacji w każdym z nowych liści dodanych w poprzednim kroku. Klasycznie do wyceny tej wykorzystuje się metodę Monte Carlo.

4. **Wsteczna propagacja wyników** uaktualnia statystyki w każdym przodku nowo dodanych liści, uwzględniając uzyskane w poprzednim kroku wyceny tych liści.

Kluczowym pomysłem w algorytmie MCTS jest zastosowanie formuły UCB. Żeby przeszukiwanie dostarczyło jak najwięcej informacji, algorytm musi budować drzewo w sposób zrównoważony. Bardziej musi się zagłębiać w dobrze rokujące warianty, ale nie może też zapomnieć o sprawdzaniu mało zbadanych zagrań, choćby wydawały się słabsze. Na etapie selekcji algorytm bierze pod uwagę średnie wyniki losowych rozgrywek dla każdego z dzieci rozważanego węzła. Formuła nie poleca jednak wybrać po prostu dziecka o nawyższym średnim wyniku. Dla każdego dziecka do średniego wyniku dodaje składnik $c_{t,s} = \sqrt{\frac{2 \ln t}{s}}$, gdzie t oznacza numer dokonowywanej iteracji, a s liczbę już dokonanych odwiedzin tego dziecka. Formuła wybiera to dziecko, dla którego opisana suma jest największa. Dzięki składnikowi $c_{t,s}$ ruchy, które nie były jak dotąd głęboko rozważone, mają większe szanse, żeby zostać wybrane (im późniejsza iteracja, tym bardziej wzrastają te szanse). Rozsądna budowa drzewa gry pozwala algorytmowi MCTS znaleźć dobre zagrania nawet, kiedy liczba dostępnych możliwości

jest duża. Przeszukiwanie drzewa gry pomaga przezwyciężyć jeden z głównych kłopotów programów opartych o czystą metodę Monte Carlo, czyli słabość taktyczną [5].

Niedługo po opublikowaniu artykułu prezentującego algorytm MCTS nowe programy (MoGo, CrazyStone) wykorzystujące tę technikę osiągnęły poziom średniozaawansowanych amatorów. W następnych latach przeprowadzano liczne eksperymenty, udostępniając programom korzystającym z MCTS bazę kształtów oraz różne elementy wiedzy dziedzinowej. W 2015 roku najsilniejsze programy grające osiągały poziom silnych amatorów. Niedługo miał nastąpić kolejny przełom.

3.2. Architektura i trening AlphaGo

Pomysł zastosowania sieci neuronowych w komputerowym Go pojawił się w ostatnich latach XX wieku. Szczególne znaczenie mają sieci konwolucyjne, które dobrze sprawdzają się w rozpoznawaniu lokalnych wzorców i łączeniu ich w większe całości. Pierwsze sieci konwolucyjne użyte w komputerowym Go były trenowane w celu przewidywania końcowego wyniku partii [8]. Na wejściu sieci znajdowała się wybrana pozycja z partii, a wyjściem sieci był spodziewany wynik. Wskazania wytrenowanej sieci mogły posłużyć jako heurystyka przy wyborze następnego ruchu. Programy używające tej techniki grały na poziomie mało zaawansowanego amatora.

Kilkanaście lat później badacze sztucznej inteligencji z zespołów DeepMind oraz Facebooka [10, 16] osiągnęli lepsze wyniki, trenując głębokie sieci konwolucyjne na przykładach wielu partii rozegranych przez silnych graczy. W przeprowadzonych eksperymentach sieć na wyjściu przyporządkowywała prawdopodobieństwa ruchom dostępnym w danej sytuacji. Tak wytrenowane sieci potrafiły poprawnie przewidzieć ponad połowę ruchów zagranych w partiach profesjonalistów. Siła gry programów wykorzystujących te sieci nadal jednak nie mogła dorównać najlepszym programom bazującym na algorytmie MCTS. Ze względu na brak algorytmu przeszukiwania programy używające głębokich sieci konwolucyjnych były słabe taktycznie.

W styczniu 2016 roku zespół DeepMind opublikował artykuł, w którym opisał stworzony przez siebie nowy program, AlphaGo. Półtora miesiąca później AlphaGo udowodniło swoją siłę w pojedynku z jednym z czołowych graczy profesjonalnych, Lee Sedolem. Mecz składał się z pięciu partii. Program odniósł zwycięstwo 4:1.

AlphaGo w trakcie gry wykorzystywało dwie sieci konwolucyjne oraz algorytm MCTS. Pierwsza z sieci (*policy network*) służyła wartościowaniu zagrań dostępnych w danej sytuacji. Została ona wytrenowana na podstawie pozycji z gier silnych amatorów i profesjonalistów przy użyciu uczenia pod nadzorem. Sieć na wyjściu miała zwracać prawdopodobieństwo dla każdego z możliwych zagrań, że jest to ruch, który został wykonany w danej pozycji przez silnego gracza.

Trening sieci był następnie kontynuowany przy użyciu uczenia ze wzmocnieniem. Program generował kolejne partie, w których do wyboru ruchów służyły kolejne wersje sieci. Pozycje z tych partii służyły jako nowe przykłady, które przedstawiane były sieci.

Druga sieć użyta w AlphaGo (*value network*) służyła do oceny pozycji na planszy. Została ona wytrenowana na podstawie pozycji z gier rozgrywanych przez *policy network*. Z każdej gry wybierana była dokładnie jedna pozycja, żeby zapobiec przeuczeniu. Na wyjściu sieć zwracała prawdopodobieństwo, że grę wygrał gracz, który właśnie ma wykonać ruch.

Zarówno *policy network* jak i *value network* były głębokami sieciami konwolucyjnymi o trzynastu warstwach. Na każdą z warstw (z paroma wyjątkami) składały się 192 powierzchnie. Na wejściu sieci znajdowała się reprezentacja pozycji na planszy złożona z 48

powierzchni wymiarów 19×19. Powierzchnie te kodowały nie tylko ułożenie kamieni na planszy, ale również pewne niskopoziomowe własności sytuacji, np. liczby oddechów łańcuchów kamieni.

W trakcie treningu AlphaGo rozgrywało partie z samym sobą, za każdym razem wybierając ruch wskazywany przez *policy network*. Pozwoliło to na wygenerowanie w niedużym czasie wielu przykładów treningowych. Jednak w oficjalnych partiach przeciwko Lee Sedolowi program musiał pokazać najwyższy poziom. Dlatego przed wyborem każdego zagrania dokonywał on analizy pozycji przy pomocy algorytmu MCTS.

W AlphaGo formuła UCB używana na etapie selekcji została zmodyfikowana tak, aby uwzględnić wyceny ruchów proponowane przez *policy network*. Kiedy dodawano nowe liście, wybierano je zgodnie ze wskazaniami sieci. W miarę jak drzewo rosło, wskazania sieci były coraz mniej brane pod uwagę. Coraz większe znaczenie zyskiwały za to wyceny potomnych węzłów.

Na etapie ewaluacji program wykorzystywał dwa rozwiązania. Po pierwsze, przeprowadzał losową rozgrywkę przy pomocy *rollout policy*. Była to funkcja wytrenowana w podobny sposób jak *policy network*. Od tej drugiej była mniej dokładna, za to wykonywała się półtora tysiąca razy szybciej. Po drugie, w tym samym czasie wyceny pozycji w liściu dokonywała również *value network*. Jako ostateczna ocena pozycji przyjmowana była średnia wyniku partii rozegranej przez *rollout policy* oraz wyceny *value network*.

AlphaGo było programem, który łączył wiele pomysłów i rozwiązań informatycznych, które w poprzednich latach przynosiły obiecujące rezultaty w komputerowym Go. Użyte były zarówno sieci konwolucyjne, jak i losowe rozgrywki. Sieci były trenowane zarówno pod nadzorem, jak i ze wzmocnieniem. Jedna sieć wartościowała ruchy, druga oceniała pozycję na planszy. W trakcie gry wybór ruchu odbywał się przy użyciu MCTS. Niecałe dwa lata później zespół DeepMind uprościł użyte rozwiązania i stworzył jeszcze silniejszy program.

3.3. Podejście *zero*

AlphaGo stanowiło całkowite odejście od klasycznych metod używanych dawniej w programach grających. W programie nie było zaimplementowanej prawie żadnej wiedzy dziedzinowej. Jedyne informacje dodatkowe względem konfiguracji kamieni na planszy, które otrzymywała sieć neuronowa, dotyczyły kilku niskopoziomowych własności łańcuchów obecnych na planszy. Głębokie zrozumienie gry AlphaGo zyskało wskutek modyfikacji wag swoich sieci neuronowych. W treningu sieci wykorzystano zapisy ludzkich gier, z których algorytm uczenia pod nadzorem zdołał wyciągnąć dużą wiedzę goistyczną. W AlphaGo nie były użyte żadne bazy kształtów ani bazy joseki, ponieważ znajomość tych wzorców została uzyskana w trakcie treningu sieci.

Opublikowany w październiku 2017 roku artykuł *Mastering the game of Go without human knowledge* [14] przyniósł kolejny przełom. Zespół DeepMind stworzył program AlphaGo Zero, który osiągnął znacząco wyższy poziom niż AlphaGo. Podobnie jak poprzedni program bazował on na głębokich sieciach konwolucyjnych i algorytmie MCTS. Tym razem jednak do treningu sieci użyto wyłącznie uczenia ze wzmocnieniem. Zrezygnowano również z ręcznego zapewnienia sieci jakichkolwiek dodatkowych informacji dotyczących pozycji na planszy. Na początku treningu AlphaGo Zero nie wiedziało nic o Go oraz nie miało żadnych danych z zewnątrz. Trening polegał wyłącznie na rozgrywaniu partii z samym sobą i używaniu pozycji z tych partii jako przykładów przedstawianych sieci neuronowej.

W AlphaGo Zero sieć neuronowa była jedna. Była ona głębsza niż sieci użyte w AlphaGo i posiadała architekturę sieci rezydualnej. W artykule opisane są dwie wersje AlphaGo Zero,

z których jedna wykorzystywała sieć 20-blokową, a druga 40-blokową. Pierwszy blok w sieci przyjmował na wejściu reprezentację pozycji na planszy i był to blok konwolucyjny. Składała się na niego konwolucja o 256 filtrach i jądrze rozmiaru 3×3, normalizacja (*batch normalization*) oraz funkcja aktywacji (*rectifier nonlinearity*). Następne bloki były to bloki rezydualne. Na każdy z tych bloków składały się dwa bloki konwolucyjne takie jak opisany powyżej. Jednakże przed zastosowaniem funkcji aktywacji w drugim z tych bloków do 256-powierzchniowej warstwy dołączane było wejście sieci. Następnie wyjście sieci było wprowadzane jako wejście do dwóch mniejszych sieci, nazywanych głowami sieci neuronowej. Jedna głowa (*policy head*) wartościowała zagrania dostępne w danej sytuacji. Druga głowa (*value head*) oceniała obecną pozycję na planszy.

Trening *value head* wyglądał tak samo, jak w przypadku *value network* w AlphaGo. Głowa uczona była przewidywania wyniku partii, które wcześniej program rozegrał sam ze sobą. Jednak w pozostałych aspektach trening sieci w AlphaGo Zero różnił się znacząco od treningu AlphaGo.

W grach, które AlphaGo Zero rozgrywało samo ze sobą, do wyboru ruchów nie używano jedynie wskazań *policy head*. Zamiast tego były one wsparte przeszukiwaniem MCTS o ustalonej głębokości. Tymczasem w przypadku AlphaGo MCTS był wykorzystany jedynie w oficjalnych partiach. Zmieniono również szczegóły działania algorytmu MCTS. Najważniejsza zmiana dotyczyła sposobu ewaluacji liści. W AlphaGo Zero ewaluacja odbywała się wyłącznie z wykorzystaniem *value head*, bez losowej rozgrywki.

AlphaGo Zero w grach treningowach nie mogło prowadzić zbyt głębokich przeszukiwań. Podstawowym celem było bowiem uzyskanie jak największej liczby przykładów. Zespół DeepMind zdecydował, że przed każdym zagraniem będzie przeprowadzane 1600 iteracji MCTS. Informacje uzyskane dzięki przeszukiwaniu MCTS dostarczały znacznie bardziej trafnych wycen ruchów niż same wskazania *policy head*. Następnie celem treningu głowy wartościującej ruchy (*policy head*) było właśnie przewidywanie tych wycen, a nie tylko przewidywanie następnego zagrania, jak to miało miejsce w AlphaGo.

Rozwiązanie zaprezentowane przez DeepMind było proste i eleganckie. Nie wymagało żadnych danych z zewnątrz ani żadnej wiedzy dziedzinowej, dlatego zostało nazwane podejściem zero. Podejście to było uniwersalne. Jeszcze w grudniu tego samego roku zespół DeepMind wykorzystał to samo rozwiązanie do wytrenowania programów grających na ponadludzkim poziomie w szachy oraz shōgi [13].

Algorytm AlphaGo Zero doczekał się wielu reimplementacji. Bardzo popularnym programem używanym przez wielu graczy trenujących Go stała się Leela Zero [12]. Jest to projekt zapoczątkowany przez Gian-Carlo Pascuttiego i rozwijany przez społeczność informatyków i goistów na githubie. Leela Zero po kilku miesiącach pracy wielu wolontariuszy doszła do mistrzowskiego poziomu gry.

Jeszcze zanim Leela Zero zdążyła przewyższyć najsilniejszych graczy na świecie, w maju 2018 roku zespół programistów z Facebooka pochwalił się własną reimplementacją algorytmu AlphaGo Zero. Stworzony przez nich program ELF OpenGo w pojedynku przeciwko czołowym profesjonalistom z Korei nie przegrał żadnej partii. Zapisy partii ujawniły jednak zaskakującą słabość programu. Okazało się, że program posiada marne zrozumienie konceptu *drabinki*.

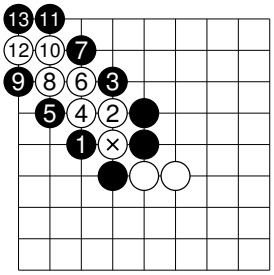


Diagram 3.1: Drabinka

Drabinka jest jedną z podstawowych technik *łapania* kamieni. Na diagramie 3.1 Czarne łapią kamień Białych ⊗ poprzez granie kolejnych atari. Ponieważ na drodze drabinki nie stoi żaden biały kamień, Białe nie mogą uciec. Kiedy drabinka dochodzi do bandy, uciekający łańcuch nie może uzyskać dodatkowych oddechów i zostaje zbity ruchem ⓭.

Jest szokujące, że program może osiągnąć ponadludzki poziom, nie potrafiąc *liczyć* drabinki. Jest to jednak ogólna słabość programów wykorzystujących podejście zero, czego dowodzi również między innymi przykład Leeli Zero. Winę za tę słabość ponosi sposób przeszukiwania drzewa. Algorytm MCTS rzadko kiedy prowadzi do rozważenia wariantu tak długiego jak drabinka.

Mimo że w partiach przeciwko profesjonalistom ELF OpenGo przed każdym ruchem wykonywał około 80 tysięcy iteracji MCTS, w zapisach partii i tak można odnaleźć błędy, które wyraźnie świadczą o nieprzeliczeniu drabinki. Najbardziej wyraźny tego przykład stanowi partia zamieszczona w dodatku B. ELF OpenGo grający Białymi *wydłużył* kamień, mając nadzieję na ucieczkę. Ruch Białych stanowi podstawowy błąd, którego uczą się unikać już gracze początkujący. Dopiero kiedy Czarne dały atari, program zorientował się, że ucieczka jest niemożliwa. W tym momencie dokonywana przez program wycena szans na zwycięstwo przechyliła się znacząco na korzyść przeciwnika. Jednakże już po kilkunastu ruchach program odrobił poniesioną stratę.

Drabinki nie stanowią jedynego słabego punktu opisanych programów. Do innych należy nieumiejętność radzenia sobie z bardzo niekorzystnymi pozycjami na planszy oraz niemożność grania z komi o innej wysokości niż użyta w trakcie treningu (w przypadku AlphaGo Zero, ELF OpenGo oraz Leeli Zero jest to komi 7,5 punktu). Niecały rok później David Wu porzucił czyste podejście zero w celu przezwyciężenia tych mankamentów oraz usprawnienia treningu. Swój program nazwał KataGo.

3.4. Podejście *semi-zero*

Podejście użyte w programie KataGo bywa czasem nazywane semi-zero. Nazwa bierze się stąd, że podobnie jak AlphaGo Zero, program uczy się wyłącznie z partii, które rozgrywa sam ze sobą. Nie ma dostępu do zapisów żadnych innych rozgrywek. Jednakże KataGo używa pewnej wiedzy dziedzinowej przy konstruowaniu wejścia sieci. Poza tym architektura sieci i sposób treningu zostały trochę zmienione z myślą o pewnych cechach gry Go. Niektóre z zastosowanych modyfikacji nie mogłyby więc zostać użyte dla innych gier takich jak szachy czy shōgi. Natomiast w przypadku Go prowadzą one do wielokrotnego przyspieszenia treningu sieci. W artykule *Accelerating Self-Play Learning in Go* [19] David Wu omawia sześć głównych wprowadzonych przez siebie modyfikacji.

1. **Manipulacja głębokością przeszukiwania MCTS w grach treningowych**
   Głowa wartościująca ruchy (*policy head*) do treningu potrzebuje pozycji z partii oraz wyniku przeszukiwania MCTS. Ważne, żeby przeszukiwanie było głębokie, aby sugerowane ruchy były znacząco lepsze niż sugestie samej sieci. Nie jest jednak konieczne, aby cała gra była rozgrywana na tak wysokim poziomie. W KataGo większość ruchów w partiach treningowych jest wybierana przy użyciu płytszego przeszukiwania, dzięki czemu partie są rozgrywane szybciej. Znacząco przyspiesza to proces nauki.

2. **Wymuszone odwiedziny liści**
   KataGo wzorem AlphaZero [13] dodaje do algorytmu MCTS czynnik losowy na etapie selekcji. Dzięki temu program jest czasami w stanie zauważyć dobry ruch, mimo że nie poleca go *policy head*. W przypadku liści dodanych dzięki zastosowaniu czynnika losowego wycena dokonywana przez *value head* często jest niska. Może się jednak okazać, że po przeliczeniu sekwencji następującej po takim ruchu, okaże się on dobry. Żeby nie przeoczyć dobrych zagrań, KataGo dla każdego z dodawanych liści wymusza pewną liczbę odwiedzin (to znaczy iteracji MCTS przechodzących przez te liście).

3. **Warstwy *global pooling***
   W Go różne globalne własności pozycji na planszy mogą mieć znaczenie dla lokalnych sytuacji. Najprostszym przykładem jest występowanie na planszy walki ko. Żeby sieć konwolucyjna miała możliwość efektywnego przekazywania tego typu informacji pomiędzy różnymi rejonami warstw (odpowiadającymi rejonom planszy), do sieci KataGo w paru miejscach dodane zostały warstwy typu *global pooling*. Warstwy te stosują globalne uśrednianie i branie globalnego maksimum dla kilku powierzchni z warstwy. Wynikowy wektor zostaje następnie dodany do pozostałych warstw.

4. **Dodatkowy cel *policy head***
   W KataGo *policy head* na wyjściu zwraca dwie powierzchnie. Pierwsza ma za zadanie przewidywać wynik przeszukiwania MCTS, tak samo jak w AlphaGo Zero. Celem drugiej jest przewidzenie odpowiedzi przeciwnika na ruch, który ma być właśnie zagrany.

5. **Dodatkowe cele *value head***
   Głowa *value head* w AlphaGo Zero zwraca tylko jedną liczbę - prawdopodobieństwo zwycięstwa. W KataGo wyjście *value head* przekazuje bogatszą informację. Sieć stara się przewidzieć również dokładny wynik partii (z uwzględnieniem różnicy punktowej). Co więcej, dla każdego przecięcia na planszy przewiduje, w czyim posiadaniu znajdzie się ono na końcu gry. Powierzchnia wyjściowa zawierająca te przewidywania nazywa się *mapą przynależności*.

6. **Bogatsze wejście sieci**
   Wejściowa warstwa w KataGo koduje nie tylko pozycję na planszy, ale również pewne nieskomplikowane własności łańcuchów kamieni. W tym względzie jest to rozwiązanie podobne do użytego w AlphaGo. W szczególności jedna z powierzchni warstwy wejściowej dla każdego przecięcia zajętego przez pewien kamień koduje informację o tym, czy łańcuch kamieni, do którego należy dany kamień, może zostać złapany w drabinkę. Sprawdzenie cech łańcuchów przekazywanych na wejściu sieci odbywa się przy pomocy prostych, szybko działających algorytmów.

Eksperymenty przeprowadzone przez Davida Wu pokazały, że każde z zastosowanych rozwiązań przyspiesza trening sieci. Najnowszy przebieg treningu KataGo (to znaczy proces treningowy rozpoczynany od zera) zakończył się w czerwcu 2020 roku. Ostateczna wersja sieci wytrenowana w tym przebiegu znacząco przewyższyła poziomem najsilniejsze sieci Leeli Zero [21]. Oprócz wymienionych zmian konstrukcja programu pozwala na grę przy użyciu różnych zestawów zasad, na planszach o różnych rozmiarach i przy różnych wartościach komi. Rozbudowane wyjście *value head* dostarcza za to bogatej informacji dotyczącej oceny sytuacji na planszy przez program. Poprzednio stworzone reimplementacje AlphaGo Zero potrafią skomentować każdy ruch jedynie przy pomocy jednej liczby oznaczjącej szanse na zwycięstwo. Sieć KataGo dodatkowo przewiduje wynik punktowy oraz końcowy kształt terytoriów obu graczy. KataGo stało się bardzo wygodnym narzędziem, z którego korzysta wielu graczy trenujących Go.

Z punktu widzenia tematu niniejszej pracy szczególnie istotne jest przewidywanie przez KataGo końcowej przynależności każdego z przecięć. Dzięki dodaniu tej funkcjonalności możliwe jest sprawdzenie, czy program ocenia lokalne sytuacje w podobny sposób jak gracze stosujący klasyczny sposób liczenia punktów.

Podsumowanie

Komputerowe Go od lat stanowiło wielkie wyzwanie dla badaczy zajmujących się sztuczną inteligencją. Było również jedną z pierwszych dziedzin, w których sprawdziły się liczne nowatorskie rozwiązania programistyczne. Metoda Monte Carlo została eksperymentalnie zastosowana w komputerowym Go jeszcze w latach dziewięćdziesiątych XX wieku. Wtedy również podejmowane były pierwsze eksperymenty związane z zastosowaniem sieci neuronowych.

Pierwszy przełom w komputerowym Go stanowiło stworzenie algorytmu przeszukiwania MCTS. Nowy sposób budowania drzewa połączony z wyceną pozycji bazującą na losowych partiach pozwoliły na przezwyciężenie taktycznej słabości programów używających czystej metody Monte Carlo.

Drugi przełom przyniosło zintegrowanie algorytmu MCTS z głębokimi sieciami konwolucyjnymi. W 2017 roku zespół DeepMind przedstawił ogólny algorytm możliwy do zastosowania we wszystkich grach o doskonałej informacji bez czynnika losowego pozwalający na trenowanie programu bez dodatku jakiejkolwiek wiedzy dziedzinowej. Siła zaproponowanego rozwiązania została dowiedziona, kiedy pozwoliło ono na osiągnięcie poziomu mistrzowskiego w Go, szachach i shōgach.

KataGo jest otwartoźródłowym projektem wykorzystującym rozwiązania zaproponowane przez DeepMind. Twórca projektu, David Wu, dodał do swojego programu kilka nowatorskich modyfikacji, które znacząco przyspieszyły trening sieci. W dodatku wprowadziły one dodatkowe funkcjonalności, dzięki którym gracz Go korzystający z programu może poznać opinię KataGo na temat tego, co się dzieje na planszy. W następnym rozdziale sprawdzam, czy KataGo, program, który przewyższył najsilniejszych profesjonalistów, dobrze rozumie zagadnienia związane z końcową fazą gry.

Rozdział 4

KataGo a końcowa faza gry

Programy wybierające ruchy na podstawie algorytmu przeszukiwania osiągają dobre wyniki w końcowej fazie gry. W przeciwieństwie do poprzednich etapów gry sytuacje, do których prowadzą sekwencje kilkunastu ruchów, mogą być wycenione bardzo precyzyjnie. Umiejętność przeliczenia setek wariantów przez program ma wówczas szczególnie dużą wartość. KataGo nie jest tutaj wyjątkiem. Siła gry programu w końcówce przewyższa poziom profesjonalistów.

W swoich badaniach skupiam się na samych wskazaniach sieci KataGo, niewspartych algorytmem MCTS. Staram się zweryfikować, czy sieć traktuje lokalne sytuacje w sposób lokalny, a jeśli tak, to czy jej wskazania odpowiadają tym wynikającym z teorii. W podrozdziale 4.1 sprawdzam, czy sieć wycenia przynależność przecięć zgodnie z teorią przedstawioną w podrozdziale 2.2. W podrozdziale 4.2 sprawdzam, czy w mało skomplikowanych sytuacjach gry końcowej *policy head* najwyżej wartościuje optymalne zagrania. W obu tych podrozdziałach testy przeprowadzam na ręcznie ułożonych przykładach pozycji umieszczonych w narożniku planszy. Dopiero w podrozdziale 4.3 sprawdzam siłę programu wspartego algorytmem przeszukiwania. Żeby wysoko zawiesić poprzeczkę, przedstawiam KataGo bardzo skomplikowaną pozycję - problem peszący zawodowców.

Żeby rezultaty eksperymentów z dwóch pierwszych podrozdziałów były miarodajne, każdy przykład badam dla wielu różnych ustawień kamieni w pozostałych częściach planszy i uśredniam uzyskane wskazania sieci. W tym celu wygenerowałem przy użyciu KataGo tysiąc losowych partii, które zaczynały się od ustalonej pozycji w lewym górnym narożniku planszy pokazanej na diagramie 4.1. Dzięki takiemu ustawieniu było wiadome, że program nigdy nie wykona ruchu w tym narożniku. Następnie do utworzenia pozycji testowych podmieniałem w zapisach partii pozycję początkową na pozycje, które zamierzałem zbadać. Układając pozycje testowe, dbałem o to, aby pozycja w narożniku z zewnątrz wyglądała tak samo jak pozycja z diagramu 4.1, to znaczy żeby z prawej strony znajdowała się linia czarnych kamieni, na dole - linia białych kamieni oraz żeby łańcuchy te miały zapewnione życie (posiadały dwoje oczu).

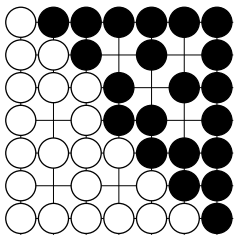


Diagram 4.1: Ustalona pozycja w lewym górnym narożniku

4.1. Testy dotyczące mapy przynależności

Mapa przynależności jest wyjściem sieci KataGo, które każdemu przecięciu na planszy przyporządkowuje liczbę rzeczywistą z przedziału $[-1,\ 1]$ reprezentującą spodziewaną przynależność przecięcia pod koniec partii. Liczba 1 oznacza przynależność do obszaru Czarnych (to znaczy, że przecięcie będzie należało do terytorium Czarnych albo będzie zajęte przez żywy czarny kamień), a -1 - że będzie należało do obszaru Białych.

W testach wskazań mapy przynależności interesująca jest wycena dokonywana na etapie partii, na którym nie wiadomo jeszcze, kto jako pierwszy zagra w danym obszarze planszy. Jako zapytania do sieci przedstawiam sytuacje powstałe w osiemdziesiątym ruchu partii. W tym momencie na planszy znajduje się jeszcze wiele ruchów o większej wartości niż ruchy w badanych przeze mnie lokalnych sytuacjach. Dlatego nawet jeśli wewnątrz sieci jest dokonywane pewne przewidywanie następnych paru ruchów, to sieć nie powinna spodziewać się, że zagranie w badanej sytuacji przez jednego z graczy jest jakkolwiek bardziej prawdopodobne niż przez drugiego. Przykładowa pozycja testowa z jednej z tysiąca wygenerowanych partii dla pierwszego testu z sekcji poświęconej ruchom z kontynuacją jest przedstawiona w dodatku C.

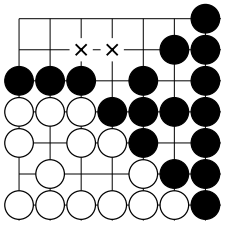
W kolejnych sekcjach przedstawiam testy dotyczące różnych zagadnień: pewnych terytoriów, ruchów bez kontynuacji, ruchów z kontynuacją, które stanowią bądź nie stanowią sente, oraz walk ko. Każde zagadnienie zbadałem na czterech przykładowych pozycjach. Dla każdej pozycji wybrałem kilka przecięć, dla których sprawdziłem wskazania sieci. Przecięcia te zaznaczam na diagramach przedstawiających badane pozycje. Przecięcia, które pod koniec partii powinny należeć do obszaru Czarnych, oznaczyłem krzyżykiem $\times$. Wskazania sieci dla tych przecięć powinny być bliskie 1. Przecięcia, które powinny należeć do obszaru Białych, oznaczyłem kółkiem $\circ$. Dla nich wskazania sieci powinny być bliskie -1. Przecięcia, które w obecnej sytuacji mogą być jednym ruchem zajęte przez Czarne albo przez Białe i ruch zajmujący takie przecięcie nie stanowi sente, oznaczyłem kwadratem $\square$. Dla takich przecięć wskazania sieci powinny być bliskie 0. Przecięcia, które można zająć w ramach kontynuacji ruchu niebędącego sente, oznaczyłem trójkątem $\triangle$. Dla takich przecięć wskazania powinny być bliskie $0,5$ lub $-0,5$ zależnie od tego, czy chodzi o kontynuację Białych, czy Czarnych. Przecięcia w sekcji poświęconej walkom ko oznaczyłem wypełnionym kwadratem $\blacksquare$. Wartości przewidywane przez teorię dla tych przecięć różnią się w zależności od przykładu.

Dla każdej pozycji testowej przedstawiam tabelę, która dla każdego z badanych przecięć przedstawia spodziewaną wartość wynikającą z teorii (W), średnią wskazań sieci (μ) dla tysiąca różnych ułożeń kamieni w pozostałych częściach planszy oraz odchylenie standardowane tych wskazań (σ).

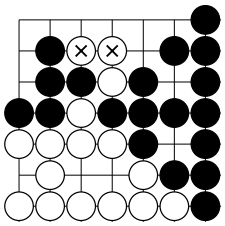
Oprócz testowania, czy wskazania sieci są bliskie wartościom wynikającym z teorii, testuję również, czy wskazania sieci są ze sobą spójne. W tym celu dla większości pozycji wybrałem więcej niż jedno przecięcie należące do tej samej kategorii. Na przykład w sekcji poświęconej pewnym terytorium na każdym diagramie krzyżykiem oznaczone są dwa przecięcia. Przewidywania dla obu przecięć należących do takiej pary powinny być takie same. Zawsze jeśli na diagramie tym samym symbolem $s \in \{\times, \circ, \square, \triangle, \blacksquare\}$ oznaczone są dwa przecięcia, to w tabeli umieszczam również kolumnę $r(s)$ przedstawiającą różnicę we wskazaniach sieci dla tych dwóch przecięć (spodziewana wartość tej różnicy jest za każdym razem równa 0).

Pewne terytoria

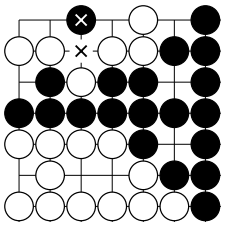
W niniejszej sekcji sprawdzam wskazania sieci dla przecięć należących do pewnego terytorium. Pierwszy przykład przedstawia pozycję, w której przynależność przecięć do terytorium Czarnych nie powinna budzić wątpliwości nawet u gracza początkującego. Następne pozycje są trochę bardziej skomplikowane i przynależność przecięć może nie być oczywista na pierwszy rzut oka. We wszystkich tych przykładach kamienie Białych w lewym górnym narożniku są martwe. Ostatni przykład jest najbardziej skomplikowany i nawet zaawansowany gracz potrzebowałby chwili, żeby upewnić się, że kamienie Białych nie mają szansy na przeżycie.



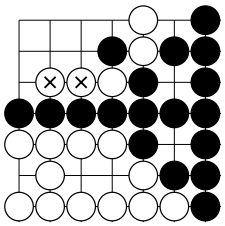
	×	×	r(×)
W	1.000	1.000	0.000
μ	0.994	0.991	0.002
σ	0.004	0.005	0.003



	×	×	r(×)
W	1.000	1.000	0.000
μ	0.908	0.918	-0.010
σ	0.041	0.038	0.018



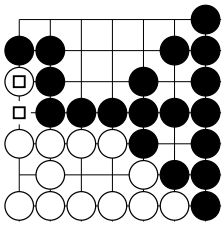
	×	×	r(×)
W	1.000	1.000	0.000
μ	0.924	0.903	0.021
σ	0.027	0.034	0.020



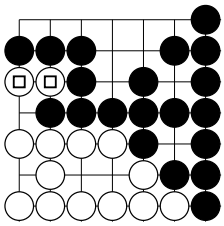
	×	×	r(×)
W	1.000	1.000	0.000
μ	0.766	0.732	0.033
σ	0.096	0.104	0.038

Ruchy bez kontynuacji

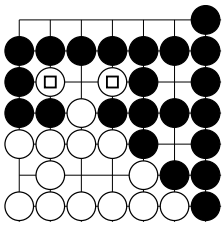
W niniejszej sekcji przedstawiam pozycje, w których o przynależności zaznaczonych przecięć zadecyduje to, który z graczy jako pierwszy wykona ruch w lokalnej sytuacji. W każdym z przykładów przynależność każdego z dwóch zaznaczonych przecięć pod koniec partii na pewno będzie taka sama. Jeśli następny ruch w danej sytuacji wykonają Czarne, to oba zaznaczone przecięcia na pewno będą pod koniec partii należały do obszaru Czarnych. Jeśli następny ruch wykonają Białe, oba przecięcia będą należały do obszaru Białych.



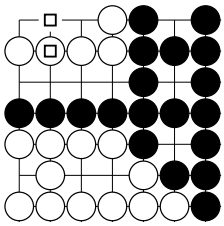
	□	□	r(□)
W	0.000	0.000	0.000
μ	0.028	0.054	-0.026
σ	0.047	0.037	0.056



	□	□	r(□)
W	0.000	0.000	0.000
μ	-0.092	-0.002	-0.089
σ	0.039	0.062	0.068



	□	□	r(□)
W	0.000	0.000	0.000
μ	-0.007	-0.051	0.045
σ	0.059	0.056	0.039

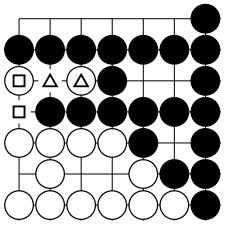


	□	□	r(□)
W	0.000	0.000	0.000
μ	-0.090	-0.046	-0.043
σ	0.157	0.150	0.068

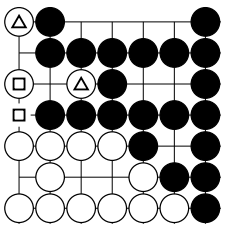
Ruchy z kontynuacją

W pozycjach badanych w tej sekcji jeśli jako pierwszy ruch wykona jeden z graczy, to granice terytoriów zostaną w pełni wyjaśnione. Natomiast jeśli to drugi gracz wykona ruch jako pierwszy, to nadal do wyjaśnienia sytuacji konieczne będzie zagranie jeszcze jednego ruchu. W pierwszych trzech przykładach sytuacja zostałaby wyjaśniona po ruchu Czarnych, a w czwartym po ruchu Białych.

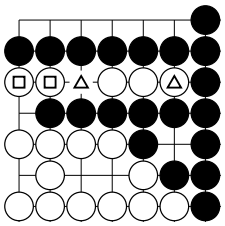
W każdej z badanych sytuacji wartość kontynuacji jest ostro mniejsza od pewnego zysku. Dlatego, zgodnie z teorią, przecięcia, które można zdobyć pierwszym ruchem, powinny być wycenione na 0, a przecięcia, które można zdobyć w ramach kontynuacji powinny być wycenione na $0,5$ (w pierwszych trzech przykładach) lub na $-0,5$ (w czwartym przykładzie). Przykład drugi różni się od pozostałych, ponieważ przynależność przecięć oznaczonych trójkątem △ może być różna pod koniec partii. Jednakże i tak przewidywania dotyczące ich przynależności powinny być takie same (po ruchu Białych ruch ratujący jeden z białych kamieni i ruch ratujący drugi z nich są równoważne).



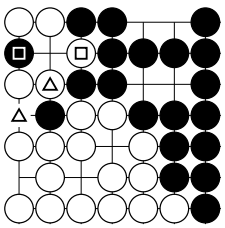
	□	□	r(□)	△	△	r(△)
W	0.000	0.000	0.000	0.500	0.500	0.000
μ	-0.144	-0.231	0.088	0.613	0.754	-0.141
σ	0.084	0.071	0.102	0.072	0.065	0.067



	□	□	r(□)	△	△	r(△)
W	0.000	0.000	0.000	0.500	0.500	0.000
μ	-0.241	-0.123	-0.119	0.663	0.641	0.023
σ	0.088	0.076	0.113	0.078	0.089	0.078



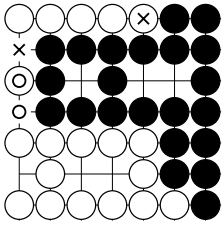
	□	□	r(□)	△	△	r(△)
W	0.000	0.000	0.000	0.500	0.500	0.000
μ	-0.254	-0.181	-0.073	0.638	0.790	-0.151
σ	0.051	0.084	0.090	0.067	0.112	0.134



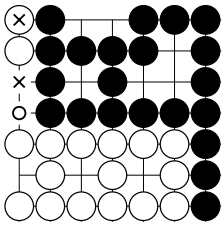
	□	□	r(□)	△	△	r(△)
W	0.000	0.000	0.000	-0.500	-0.500	0.000
μ	0.066	0.205	-0.140	-0.684	-0.784	0.100
σ	0.068	0.048	0.071	0.073	0.058	0.053

Ruchy sente

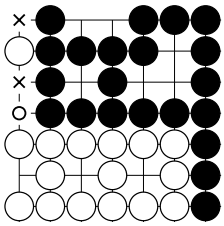
W tej sekcji przedstawiam sytuacje podobne do tych z poprzedniej sekcji z tą różnicą, że wartość kontynuacji przekracza w nich pewny zysk. Przy poprawnej grze obu graczy Białe powinny wykorzystać swój przywilej i wykonać pierwszy ruch sekwencji, kiedy największy ruch w pozostałej części planszy będzie mniejszy niż wartość kontynuacji. Wówczas Czarne powinny odpowiedzieć na ruch Białych, uniemożliwiając im zagranie kontynuacji. Dlatego przecięcia oznaczone kółkiem ○ pod koniec partii powinny znaleźć się w obszarze Białych. Z kolei przecięcia oznaczone krzyżykiem × powinny znaleźć się w obszarze Czarnych.



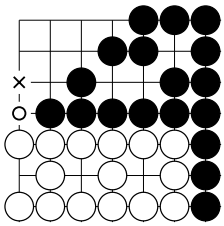
	×	×	r(×)	○	○	r(○)
W	1.000	1.000	0.000	-1.000	-1.000	0.000
μ	0.892	0.974	-0.082	-0.631	-0.509	-0.122
σ	0.046	0.013	0.049	0.112	0.060	0.068



	×	×	r(×)	○
W	1.000	1.000	0.000	-1.000
μ	0.945	0.974	-0.030	-0.423
σ	0.021	0.012	0.021	0.043



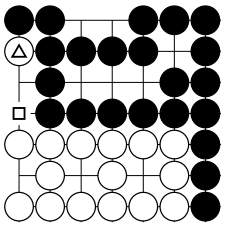
	×	×	r(×)	○
W	1.000	1.000	0.000	-1.000
μ	0.867	0.979	-0.112	-0.363
σ	0.031	0.008	0.027	0.044



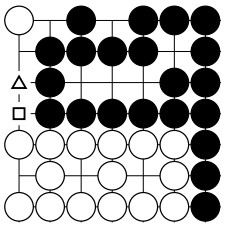
	×	○
W	1.000	-1.000
μ	0.914	-0.352
σ	0.025	0.056

Ruchy na granicy sente

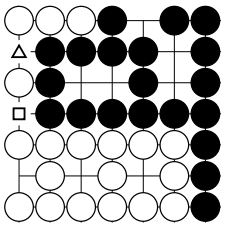
W niniejszej sekcji przedstawiam pozycje, w których pewny zysk Białych i wartość kontynuacji są sobie równe. Zgodnie z teorią zaprezentowaną w podrozdziale 2.2 ruchów w takich sytuacjach nie uznaje się za sente. W rzeczywistości kwestia ta jest jednak problematyczna[1]. Jeżeli w pewnym momencie partii taki ruch jest największy na planszy i jest akurat kolej Białych, to po ruchu Białych odpowiedź na ten ruch znowu będzie największym ruchem na planszy. W wielu partiach będzie to jedyny ruch o takiej wartości, więc Czarne od razu odpowiedzą. Dlatego spodziewane wartości W przedstawione w tabeli niekoniecznie są tymi, których należy się spodziewać w prawdziwych partiach. To, jak często zdarza się, że gracz od razu odpowiada na ruch stanowiący przypadek graniczny w definicji sente, mogłoby być przedmiotem badań statystycznych. Niniejsza sekcja w tej pracy stanowi raczej pewną ciekawostkę i może otwierać pole do dalszych badań.



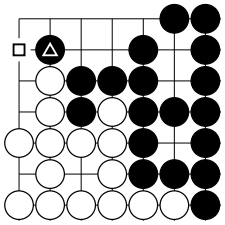
	□	△
W	0.000	0.500
μ	-0.351	0.913
σ	0.042	0.020



	□	△
W	0.000	0.500
μ	-0.304	0.681
σ	0.041	0.056



	□	△
W	0.000	0.500
μ	-0.363	0.813
σ	0.073	0.048

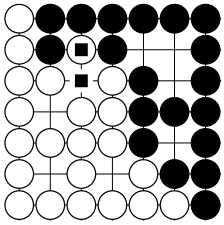


	□	△
W	0.000	0.500
μ	-0.379	0.811
σ	0.068	0.056

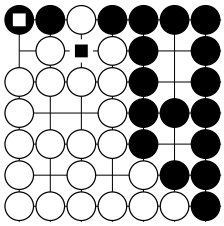
[1] Obliczenie wartości takich ruchów nie budzi wątpliwości. Niezależnie, czy traktować taki ruch jako sente, czy nie, jego wartość będzie równa pewnemu zyskowi. Jednakże te dwie różne interpretacje dają inne przewidywania ostatecznej przynależności przecięć: albo 0 i $0,5$, albo -1 i 1 odpowiednio dla przecięć oznaczonych kwadratem i trójkątem.

Walki ko

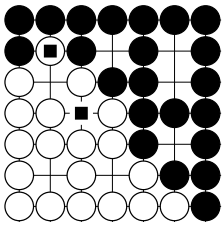
Niniejsza sekcja przedstawia pozycje, w których o ostatecznej przynależności przecięć zadecyduje walka ko. Diagramy przedstawiają kolejno: zwykłe ko o jeden punkt (które często występuje pod koniec partii), zwykłe ko o większą liczbę punktów oraz tak zwane dwupoziomowe ko i dwustopniowe ko. Kwestia obliczania lokalnych wyników dla takich sytuacji nie została omówiona w niniejszej pracy. Zainteresowany czytelnik może jednak spróbować sam rozważyć to zagadnienie. Pomocna okaże się tutaj ta sama metoda, której użyliśmy na diagramie 2.2, to znaczy ustawienie kilku takich samych sytuacji obok siebie. W przypadku zwykłego ko konieczne jest ustawienie trzech takich samych sytuacji.



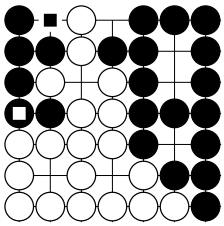
	■	■	r(■)
W	-0.333	-0.333	0.000
μ	-0.581	-0.447	-0.134
σ	0.083	0.052	0.091



	■	■	r(■)
W	-0.333	-0.333	0.000
μ	-0.291	-0.353	0.063
σ	0.062	0.060	0.075



	■	■	r(■)
W	-0.500	-0.500	0.000
μ	-0.362	-0.780	0.418
σ	0.101	0.044	0.078



	■	■	r(■)
W	-0.556	-0.556	0.000
μ	-0.521	-0.459	-0.062
σ	0.079	0.105	0.063

4.2. Testy dotyczące wskazań *policy head*

Policy head jest głową sieci KataGo, na wyjściu której każdemu niezajętemu przecięciu (o ile ruch na tym przecięciu jest dozwolony) oraz pasowi zostaje przyporządkowana wartość z przedziału [0, 1]. Wartości te sumują się do jedności. Wskazania *policy head* są używane w algorytmie MCTS na etapie selekcji. Im wyższa wartość jest przyporządkowana danemu przecięciu, tym większa szansa, że algorytm przeszukiwania w kolejnej iteracji wybierze wariant rozpoczynający się od tego ruchu. W trakcie treningu ze wzmocnieniem *policy head* uczona jest przewidywania wyników przeszukiwań MCTS, których program dokonał w trakcie partii. Można więc powiedzieć, że *policy head* dla każdego przecięcia przewiduje szanse, że jest to najlepszy dostępny ruch.

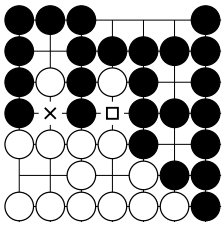
Testy dotyczące wskazań *policy head* przeprowadzam dla pozycji końcowych wygenerowanych partii, w których pozycja w lewym górnym narożniku (przedstawiona na diagramie 4.1) została zastąpiona przez pozycje, w których pozostały tylko dwie niewyjaśnione lokalne sytuacje. W każdej z tych dwóch lokalnych sytuacji zanaczyłem ruch, który powinien zostać zagrany jako następny. W większości przykładów jeden z tych ruchów jest poprawny, a drugi nie (to znaczy prowadzi do straty punktowej). Poprawne ruchy na diagramach oznaczam krzyżykiem $\times$, a niepoprawne kwadratem $\square$. W kilku przykładach oba zaznaczone ruchy prowadzą do tego samego rezultatu punktowego, a zatem oba są poprawne. Takie ruchy oznaczam trójkątem $\triangle$. Dla każdej pozycji zamieszczam analogiczną tabelę statystyk jak w przypadku testów mapy przynależności.

Dla każdego badanego przykładu przeprowadziłem również drugi test, w którym lokalne sytuacje zawierające każdy z możliwych do wykonania ruchów, umieszczone są w przeciwległych narożnikach. W ten sposób sprawdzam, czy bliskość sytuacji, w których znajdują się możliwe do wykonania ruchy, wpływa na analizę pozycji na planszy dokonywaną przez sieć KataGo. Żeby przeprowadzić te testy, wygenerowałem kolejne tysiąc partii, w których tym razem na pozycję początkową składało się ułożenie kamieni przedstawione na diagramie 4.1 umieszczone zarówno w lewym górnym jak i w prawym dolnym narożniku planszy. Dla testów przeprowadzonych w ten sposób utworzyłem analogiczne tabele statystyk, które umieściłem po prawej stronie tabel podsumowujących pierwsze testy. Przykładowe całoplanszowe pozycje użyte w obydwu wersjach testów dla pierwszego przykładu z pierwszej sekcji przedstawione są w dodatku D na diagramie D.1.

W pierwszych dwóch sekcjach przedstawiam przykłady ilustrujące ruchy o różnych wartościach (z kontynuacją lub bez), dla których większy ruch jest tym poprawnym. W trzeciej sekcji przedstawiam przykłady, w których każdy z ruchów prowadzi do tego samego rezultatu punktowego. W czwartej sekcji przedstawiam przykłady, w których poprawny jest ruch o mniejszej wartości ze względu na tedomari. W ostatniej sekcji skupiam się na sytuacjach omówionych w podrozdziale 2.5, w których należy dokonać wyboru pomiędzy jednopunktowymi ruchami wchodzącymi w różne korytarze. Dla tych sytuacji inaczej konstruuję testy z pozycjami ustawionymi w dwóch narożnikach. Pozycje ustawiane w przeciwległym narożniku służą mi do stworzenia całoplanszowej pozycji, w której niepoprawna kolejność wchodzenia w korytarze doprowadzi do straty jednego punktu. W dodatku D umieszczone są przykładowe całoplanszowe pozycje dla pierwszego testu dotyczącego korytarzy (diagram D.2).

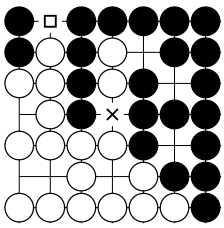
Ruchy o różnych wartościach

W niniejszej sekcji przedstawiam przykłady, w których dostępne są dwa ruchy, spośród których żaden nie posiada kontynuacji i jeden zapewnia większy zysk punktowy niż drugi. W takich sytuacjach zawsze poprawny jest wybór ruchu o większej wartości.



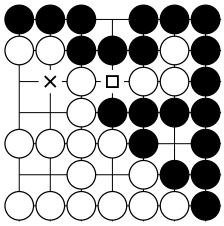
Jeden narożnik		
	×	□
μ	0.584	0.147
σ	0.119	0.047

Dwa narożniki		
	×	□
μ	0.638	0.112
σ	0.133	0.047



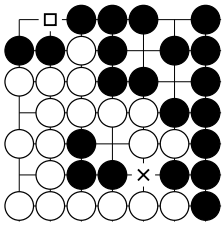
Jeden narożnik		
	×	□
μ	0.473	0.266
σ	0.111	0.056

Dwa narożniki		
	×	□
μ	0.588	0.188
σ	0.141	0.065



Jeden narożnik		
	×	□
μ	0.530	0.287
σ	0.090	0.083

Dwa narożniki		
	×	□
μ	0.531	0.292
σ	0.099	0.072

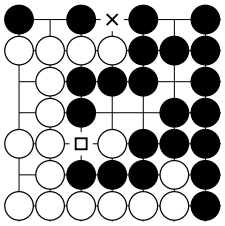


Jeden narożnik		
	×	□
μ	0.538	0.190
σ	0.152	0.114

Dwa narożniki		
	×	□
μ	0.542	0.246
σ	0.126	0.122

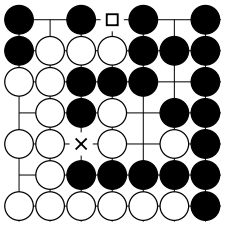
Ruchy z kontynuacją o różnych wartościach

W niniejszej sekcji przedstawiam przykłady, w których w jednej lokalnej sytuacji to ruch Czarnych posiada kontynuację, a w drugiej to ruch Białych posiada kontynuację. W pierwszym przykładzie pewne zyski tych ruchów są równe, jednak kontynuacja jednego jest większa niż drugiego. W pozostałych przykładach pewny zysk jednego z ruchów jest większy. Poza tym maksymalny zysk (czyli pewny zysk plus dwukrotność wartości kontynuacji) możliwy do odniesienia z tego ruchu też jest większy niż maksymalny zysk możliwy do odniesienia z drugiego.



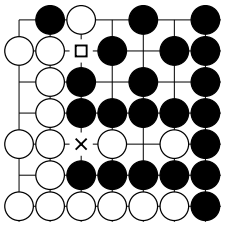
Jeden narożnik		
	×	□
μ	0.489	0.276
σ	0.100	0.067

Dwa narożniki		
	×	□
μ	0.514	0.260
σ	0.109	0.065



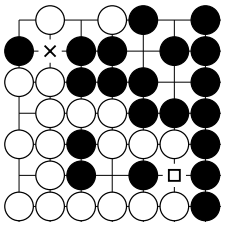
Jeden narożnik		
	×	□
μ	0.534	0.295
σ	0.129	0.094

Dwa narożniki		
	×	□
μ	0.551	0.287
σ	0.143	0.094



Jeden narożnik		
	×	□
μ	0.431	0.335
σ	0.100	0.073

Dwa narożniki		
	×	□
μ	0.428	0.360
σ	0.103	0.088

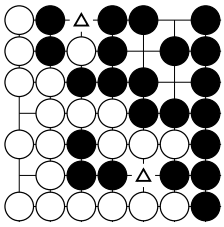


Jeden narożnik		
	×	□
μ	0.584	0.229
σ	0.158	0.109

Dwa narożniki		
	×	□
μ	0.607	0.218
σ	0.164	0.109

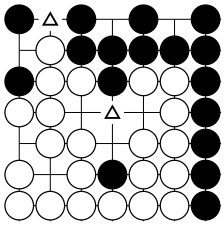
Ruchy prowadzące do tego samego rezultatu

W niniejszej sekcji przedstawiam pozycje, w których wykonanie każdego z dwóch możliwych ruchów prowadzi do tego samego rezultatu punktowego. W pierwszych dwóch przykładach są to lokalne sytuacje, dla których drzewa gry (po usunięciu zdominowanych opcji i skróceniu gałęzi) są identyczne. Ruchy w tych sytuacjach są więc równoważne z punktu widzenia kombinatorycznej teorii gier. W pozostałych dwóch przykładach drzewo gry jednej z sytuacji stanowi lustrzane odbicie drzewa gry drugiej - w jednej to ruch Czarnych posiada kontynuację, a w drugiej ruch Białych. Ponieważ są to jedyne dwie nierozstrzygnięte sytuacje na planszy, to ruch w dowolnej z nich prowadzi do tego samego rezultatu punktowego. Trzecia pozycja stanowi przykład zaadaptowany z trzeciego rozdziału książki *Mathematical Go* [4].



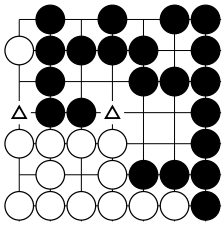
Jeden narożnik		
	△	△
μ	0.373	0.396
σ	0.091	0.075

Dwa narożniki		
	△	△
μ	0.439	0.407
σ	0.081	0.078



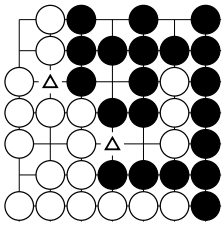
Jeden narożnik		
	△	△
μ	0.548	0.250
σ	0.121	0.075

Dwa narożniki		
	△	△
μ	0.574	0.256
σ	0.121	0.076



Jeden narożnik		
	△	△
μ	0.276	0.267
σ	0.044	0.051

Dwa narożniki		
	△	△
μ	0.274	0.286
σ	0.053	0.042



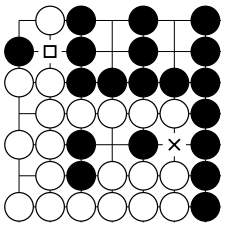
Jeden narożnik		
	△	△
μ	0.471	0.262
σ	0.095	0.060

Dwa narożniki		
	△	△
μ	0.468	0.294
σ	0.097	0.064

Tedomari

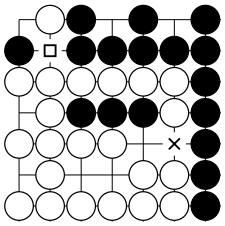
W niniejszej sekcji badam sytuacje, w których obliczenie wartości ruchów nie zapewnia wyboru poprawnego zagrania. Na przedstawionych przykładach jeden z ruchów posiada kontynuację, a drugi nie. W pierwszych dwóch przykładach ruch z kontynuacją ma mniejszą wartość, jednak to jego należy wybrać. Przedstawione lokalne sytuacje są jedynymi niewyjaśnionymi sytuacjami na planszy i Czarne odniosą większy zysk, jeśli wybiorą ten mniejszy ruch, a następnie zagrają również jego kontynuację.

Dwa pozostałe przykłady dotyczą wyboru spośród ruchów o tej samej wartości. Czarne mogą zagrać albo jednopunktowy ruch, który nie ma kontynuacji dla żadnego z graczy, albo jednopunktowy ruch, który dla Białych miałby jednopunktową kontynuację. Zgodnie z ustaleniami z podrozdziału 2.2 wybór należy uzależnić od tego, czy liczba pozostałych na planszy jednopunktowych ruchów bez kontynuacji jest parzysta. W trzecim przykładzie taki ruch jest tylko jeden i dlatego należy wybrać ten właśnie ruch. W czwartym przykładzie istnieją trzy dostępne ruchy, spośród których dwa to jednopunktowe ruchy bez kontynuacji. W wersji testu z sytuacjami w przeciwległych narożnikach, dostępnych ruchów jest aż pięć i cztery z nich to jednopunktowe ruchy bez kontynuacji. Ponieważ liczba jednopunktowych ruchów bez kontynuacji jest parzysta, to poprawny wybór stanowi ruch, który dla Białego posiadałby kontynuację.



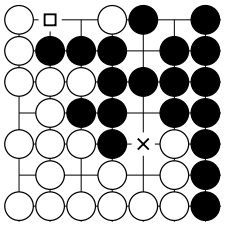
Jeden narożnik		
	×	□
μ	0.523	0.321
σ	0.107	0.101

Dwa narożniki		
	×	□
μ	0.573	0.297
σ	0.106	0.089



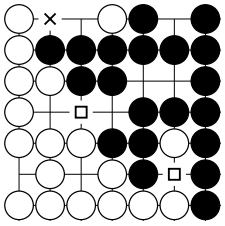
Jeden narożnik		
	×	□
μ	0.416	0.429
σ	0.090	0.092

Dwa narożniki		
	×	□
μ	0.444	0.422
σ	0.092	0.098



Jeden narożnik		
	×	□
μ	0.599	0.101
σ	0.133	0.041

Dwa narożniki		
	×	□
μ	0.627	0.112
σ	0.146	0.052



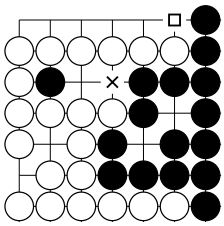
Jeden narożnik			
	×	□	□
μ	0.222	0.251	0.300
σ	0.036	0.033	0.053

Dwa narożniki					
	×	□	□	□	□
μ	0.171	0.178	0.163	0.162	0.151
σ	0.043	0.021	0.020	0.024	0.017

Wchodzenie w korytarze

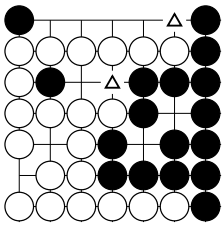
W niniejszej sekcji badam zaawansowane zagadnienie kolejności wchodzenia w korytarze. Poprawny wybór zależy od wartości ruchów na końcu korytarzy oraz od długości korytarzy. W pierwszym przykładzie jeden z korytarzy jest zakończony ruchem jednopunktowym, a drugi mniejszym. Należy wchodzić w pierwszy korytarz. W drugim przykładzie oba korytarze zakończone są jednopunktowym ruchem i kolejność wchodzenia nie ma znaczenia. Dlatego oba ruchy są oznaczone trójkątem △. W trzecim przykładzie przedstawione korytarze są zakończone ruchem o tej samej wielkości, większym niż jeden punkt. Poprawny jest ruch wchodzący w krótszy korytarz. W czwartym przykładzie korytarze są zakończone ruchami różnej wielkości przekraczającymi jeden punkt. Poprawny wybór to wchodzenie w korytarz zakończony mniejszym ruchem.

W przedstawionych pozycjach wybór niepoprawnego ruchu nie prowadzi do straty punktowej. Jednakże, jak było pokazane w podrozdziale 2.5, istnieją sytuacje, w których ruch niepoprawny prowadzi do straty. Sytuacje te badam w wersji testu z pozycjami umieszczonymi w przeciwległych narożnikach. W drugim narożniku umieszczam te same pozycje, tylko z zamienionymi kolorami, w których Białe wykonały już jeden ruch wchodzący w jeden z korytarzy Czarnych. Jest to analogiczna konstrukcja przykładów jak w przypadku diagramu 2.19. W każdym z przypadków ruchy w przeciwległym narożniku blokujące jeden z własnych korytarzy są niepoprawne i prowadzą do straty punktowej. Do straty punktowej prowadzi również niepoprawny wybór korytarza Białych, w który mają wejść Czarne.



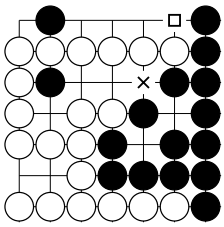
Jeden narożnik		
	×	□
μ	0.428	0.341
σ	0.082	0.050

Dwa narożniki				
	×	□	□	□
μ	0.341	0.211	0.191	0.095
σ	0.090	0.036	0.041	0.025



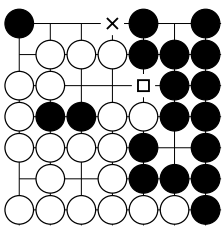
Jeden narożnik		
	△	△
μ	0.422	0.343
σ	0.081	0.051

Dwa narożniki				
	△	△	□	□
μ	0.335	0.216	0.187	0.101
σ	0.085	0.033	0.040	0.025



Jeden narożnik		
	×	□
μ	0.386	0.363
σ	0.064	0.054

Dwa narożniki				
	×	□	□	□
μ	0.297	0.115	0.232	0.177
σ	0.063	0.026	0.031	0.029



Jeden narożnik		
	×	□
μ	0.379	0.362
σ	0.061	0.055

Dwa narożniki				
	×	□	□	□
μ	0.274	0.261	0.094	0.198
σ	0.047	0.040	0.024	0.037

4.3. KataGo a problem peszący zawodowców

W ramach ostatniego testu sprawdziłem, jak KataGo poradzi sobie z problemem peszącym zawodców. Tym razem przyjrzałem się, jakie ruchy proponować będzie program po wykonaniu wielu iteracji algorytmu MCTS. Problem przedstawiony jest na diagramie 4.2.

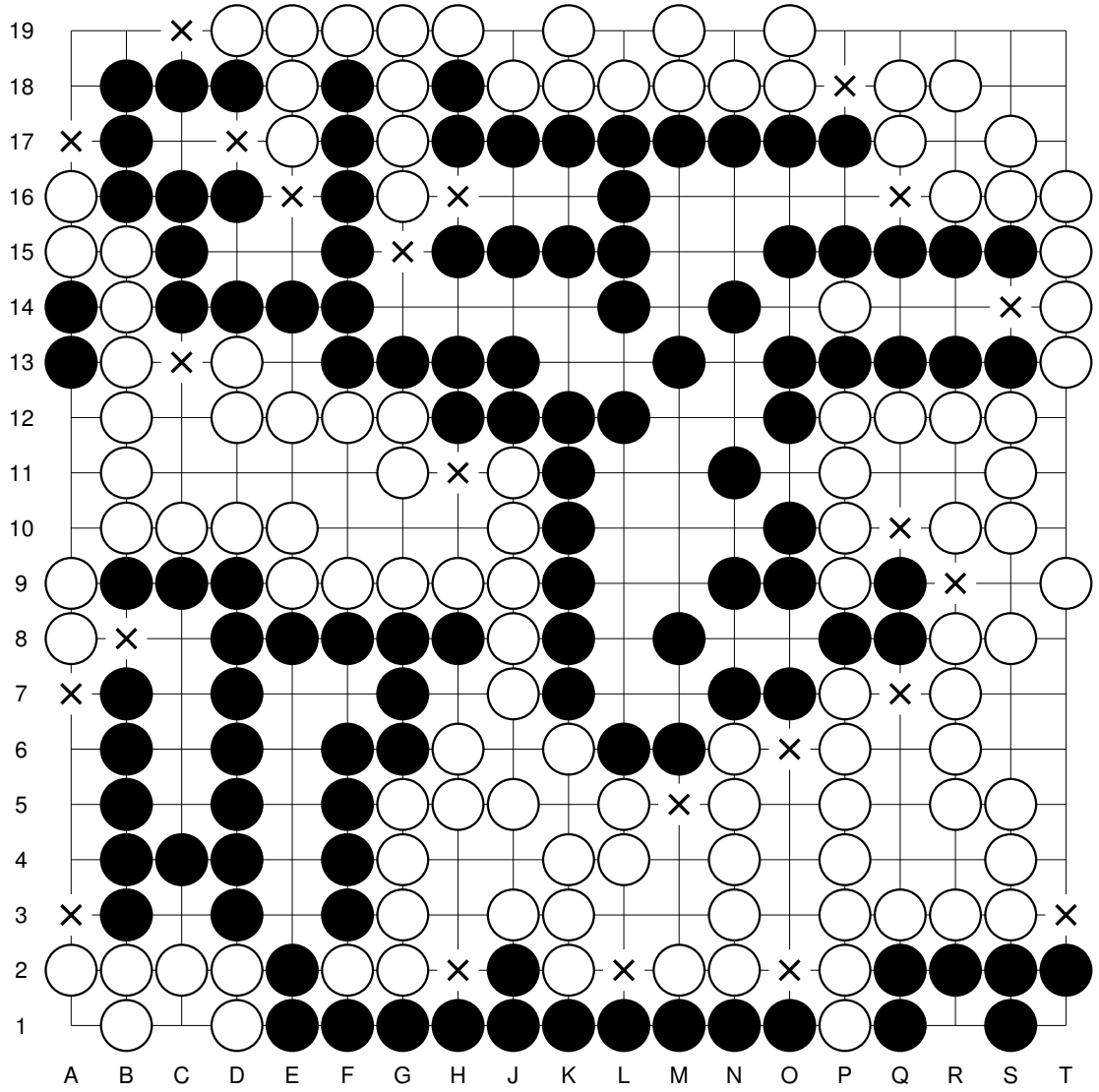


Diagram 4.2: Problem peszący zawodowców: ruch Białych, komi = 0

W przedstawionej sytuacji Białe mają dwadzieścia trzy dostępne zagrania (zaznaczone krzyżykami), z których każde jest warte około jednego punktu. Klasyczny sposób liczenia wartości ruchów wystarczy, żeby stwierdzić, że ruchy o współrzędnych P18, Q16, S14, T3 są warte jeden punkt, a ruchy C13 i H11 są trochę mniejsze. W czterech częściach planszy znajdują się niepołączone łańcuchy mające dostęp do wielu korytarzy: na górnej, lewej i dolnej bandzie oraz w centrum po prawej stronie. Do poprawnej wyceny siedemnastu z dostępnych ruchów (o współrzędnych Q11, R9, Q7, H2, L2, O2, M5, O6, A3, A7, B8, A17, C19, D17, E16, G15, H16) konieczna jest analiza zaprezentowana pod koniec podrozdziału 2.5. Zgodnie z tą analizą tylko ruchy Q11, R9 i Q7 mają wartość jednego punktu. Pozostałe są nieznacznie mniejsze.

Zgodnie z ustaleniami z podrozdziału 2.5 poprawny ruch Białych w przedstawionej sytuacji to S14, ponieważ jest to ruch, który wchodzi w korytarz zakończony najmniejszym ruchem o wartości powyżej jednego punktu. Bliższa analiza pokazuje, że ruchy Q16, Q10, R9, Q7 również nie muszą prowadzić do straty punktowej, o ile tylko Białe wkrótce zagrają ruch S14 (i następnie R14). Rozpoczęcie gry od jakiegokolwiek innego ruchu prowadzi do straty jednego punktu i zakończenia partii remisem. Poniżej przedstawiam ruchy, które w tej sytuacji proponuje KataGo. W każdej z tabel ruch, który program uznaje za najlepszy, umieszczony jest w kolumnie z lewej strony. Pozostałe kolumny zawierają ruchy, które program uznaje za następne w kolejności dobrych zagrań.

Wskazania *policy head* (bez iteracji MCTS):

Ruch	O2	L2	H2	Q16	Q10	S14	P18	G15
Wskazania *policy head*	0.190	0.143	0.102	0.101	0.068	0.059	0.058	0.056

Wskazania KataGo po 100 iteracjach MCTS:

Ruch	G15	H2	Q16	O2	S14	P18	H11	B8
Liczba wizyt	33	19	15	11	9	5	3	3

Wskazania KataGo po 1000 iteracjach MCTS:

Ruch	O2	Q16	G15	L2	H2	S14	P18	Q7
Liczba wizyt	311	192	126	68	58	51	49	22

Wskazania KataGo po 10000 iteracjach MCTS:

Ruch	G15	Q16	O2	H2	B8	S14	L2	P18
Liczba wizyt	7111	2083	715	563	531	427	270	192

Wskazania KataGo po 100000 iteracjach MCTS:

Ruch	B8	G15	Q16	O2	S14	H2	L2	P18
Liczba wizyt	78841	37730	9321	5058	3982	2246	2176	1793

Wskazania KataGo po 1000000 iteracjach MCTS:

Ruch	G15	Q16	O2	S14	B8	H2	L2	P18
Liczba wizyt	797506	65229	22185	13338	11564	9681	8738	7067

Rozdział 5

Interpretacja wyników i podsumowanie

Testy pokazały, że KataGo do pewnego stopnia rozumie zagadnienia końcowej fazy gry opisane w rozdziale 2. Zrozumienie to nie jest jednak doskonałe. W podrozdziale 5.1 analizuję wyniki testów opisanych w podrozdziale 4.1, w podrodziale 5.2 skupiam się na testach opisanych w podrozdziale 4.2, a w podrozdziale 5.3 komentuję wyniki testu opisanego w podrozdziale 4.3.

5.1. Dokładność mapy przynależności

KataGo rozszerza algorytm użyty w AlphaZero o kilka dodatkowych rozwiązań usprawniających trening sieci. Jednym z usprawnień jest bogatsze wyjście sieci, na które między innymi składa się mapa przynależności. Podobnie jak wszystkie inne programy reimplementujące algorytm AlphaZero, KataGo gra, wybierając ruchy przy użyciu algorytmu MCTS wspartego o wskazania sieci. Na etapie selekcji wykorzystywane jest wyjście *policy head* przyporządkowujące każdemu dostępnemu zagraniu prawdopodobieństwo, że jest to najlepszy ruch w danej sytuacji. Na etapie ewaluacji wykorzystywane jest wyjście *value head* oceniające szanse na zwycięstwo dla aktualnej sytuacji na planszy. Zgodnie z artykułem *Accelerating Self-Play Learning in Go* [19] dodanie do funkcji straty składnika odpowiedzialnego za dokładność przewidywań mapy przynależności poprawia regularyzację. Jednakże mapa przynależności nie jest wykorzystywana przez program w trakcie gry. Dlatego prawdopodobnie nie powinno dziwić, że przewidywania dotyczące przynależności przecięć nie są bardzo precyzyjne.

W podrozdziale 4.1 przedstawiłem wyniki testów mapy przynależności. Pierwsza sekcja pokazała, że KataGo z dużą pewnością przyporządkowuje graczowi przecięcia należące do jego pewnego terytorium. Przykłady z martwymi kamieniami przeciwnika pokazały również głębokie zrozumienie kształtów zaszyte w sieci KataGo. Nawet w ostatnim, najbardziej skomplikowanym przykładzie sieć wskazuje, że przecięcia, na których stoją kamienie Białych, w rzeczywistości należą do obszaru Czarnych. Jednakże wskazanie sieci mimo wszystko nie jest tak bliskie poprawnej wartości jak w przypadku terytorium bez kamieni o niepewnym statusie (około $0,75$, a nie $0,99$).

Wskazania mapy przynależności można by chcieć wykorzystać w celu oszacowania wyniku partii. Gdyby były one bardzo precyzyjne, to suma wskazań dla wszystkich przecięć na planszy byłaby bliska spodziewanemu wynikowi. Skoro jednak nawet w przypadku pewnych terytoriów wskazania sieci bywają odległe od poprawnej wartości o dużą część punktu, to suma wskazań nie może stanowić bardzo precyzyjnego oszacowania końcowego rezultatu.

Testy w drugiej sekcji pokazują, że dla ruchów bez kontynuacji sieć dobrze ocenia przynależność przecięć. Poprawna wycena w tych przypadkach to 0. Średnie predykcji sieci mieszczą się w przedziale $(-0,1,\ 0,1)$, a odchylenia standardowe nie przekraczają $0,06$.

W przypadku ruchów z kontynuacją wskazania sieci zaczynają odbiegać od przewidywań teorii. Kiedy pewny zysk z ruchu przekracza wartość kontynuacji, ruch nie jest uznawany za sente. Przynależność przecięć, które można zdobyć tym ruchem, powinna być wyceniona na 0, co oznacza, że z równym prawdopodobieństwem będą one pod koniec partii należały do Czarnych albo do Białych. Sieć wycenia jednak, że są trochę większe szanse, że przecięcia takie będą należały do gracza, którego ruch posiada kontynuację. Można rzec, że ocenia takie ruchy jako *prawie sente*. W drugim przykładzie w sekcji drugiej błędne przewidywanie sieci można uznać za szczególnie rażące, ponieważ ruch w tym przykładzie tylko z pozoru posiada kontynuację. W rzeczywistości po tym ruchu powstają dwie możliwe kontynuacje o tej samej wartości, co jest równoważne sytuacji, w której żadna kontynuacja by nie istniała.

Mimo wszystko sieć rozumie do pewnego stopnia pojęcie sente. Wskazania są inne, kiedy pewny zysk przekracza wartość kontynuacji, inne, kiedy pewny zysk i wartość kontynuacji są sobie równe, a jeszcze inne, kiedy to wartość kontynuacji jest większa. Rozgraniczenia pomiędzy tymi kategoriami nie są jednak całkiem wyraźne. Za szczególnie rażący błąd można uznać przewidywania dla ostatniego przykładu w sekcji poświęconej sente. Ruch Białych na pierwszej linii stanowi absolutny przywilej: pewny zysk to zaledwie jeden punkt, a kontynuacja jest warta cztery punkty. Partie, w których Białe nie zdążyłyby wykorzystać tego typu przywileju zdarzają się bardzo rzadko. Tymczasem sieć ocenia przynależność przecięcia oznaczonego kółkiem średnio na $-0,352$ zamiast na -1.

Ostatnia sekcja poświęcona walkom ko jest znacznie trudniejsza w interpretacji, ponieważ wartości przewidywane przez teorię mogą się różnić od tego, do którego z graczy w prawdziwych partiach statystycznie będą należały przecięcia. Może być tak zwłaszcza dla dwustopniowego ko pokazanego w ostatnim przykładzie. Dokładniejsza dyskusja tego zagadnienia wykracza poza zakres niniejszej pracy. Pouczające są natomiast dwa pierwsze przykłady przedstawiające zwykłe ko. Zgodnie z teorią, gracz, który w danym momencie ma zbite ko, posiada szanse w wysokości $\frac{2}{3}$ na jego wygranie. W przypadku ko o sześć punktów (drugi przykład) wskazania sieci są bliskie przewidywanej przez teorię wartości $(-0,333)$. Jednakże dla ko o jeden punkt (takiego jak w pierwszym przykładzie) ze statystycznego punktu widzenia szanse na jego wygranie są inne. Takie ko jest zawsze rozgrywane jako ostatnia rzecz w partii. Statystycznie w połowie przypadków gracz, który ma zbite ko, będzie lokalnie wykonywać ruch jako pierwszy i je połączy. W pozostałych przypadkach każdy z graczy powinien mieć statystycznie 50% szans na wygranie ko (zwycięży ten, kto będzie miał na planszy więcej gróźb). Dlatego gracz, który ma zbite ko o jeden punkt, posiada $\frac{3}{4}$ szans na jego wygranie. Przynależność przecięć w pierwszym przykładzie należałoby więc wycenić na $-0,5$. Wskazania sieci faktycznie skupiają się raczej wokół tej wartości, a nie wokół $-0,333$.

Oprócz tego we większości przykładów sprawdzałem również spójność wskazań KataGo. Aby to sprawdzić, porównywałem wskazania dla par przecięć, które na pewno znajdą się pod koniec partii w obszarze tego samego gracza. Wskazania sieci dla takich przecięć powinny być więc takie same. Zazwyczaj różnica ta była jednak równa około $0,05$, a w niektórych przykładach przekraczała nawet $0,1$.

Podsumowując, KataGo poprawnie ocenia przynależność przecięć należących do pewnych terytoriów oraz takich, które każdy z graczy może zdobyć jednym ruchem nieposiadającym kontynuacji. Sieć posiada również pewne zrozumienie konceptu sente i uzależnia swoje wskazania od tego, czy większy jest pewny zysk, czy też wartość kontynuacji. Jednakże wskazania te nie są już całkiem dokładne. Na podstawie porównania różnic we wskazaniach dla par przecięć należących do tej samej kategorii można powiedzieć, że wszystkie wskazania sieci ce-

chują się nieostrością na poziomie około $0,05$ punktu. Być może dodatkowy trening skupiony konkretnie na przewidywaniu przynależności przecięć w lokalnych sytuacjach gry końcowej mógłby jeszcze trochę wzmocnić sieć KataGo. Podejrzewam, że szczególnie dużą korzyść sieć mogłaby odnieść, gdyby w wyniku takiego treningu bardziej klarowne stały się przewidywania przynależności przecięć w sytuacjach z ruchami posiadającymi kontynuację i sieć bardziej wyraźnie zaczęłaby odróżniać ruchy stanowiące i niestanowiące sente.

5.2. Ruchy proponowane przez *policy head*

Przykłady w pierwszych dwóch sekcjach przedstawiały ruchy o różnych wartościach. Granice terytoriów były zawsze wyjaśnione po jednym ruchu albo po ruchu i jego kontynuacji. Poprawny był zawsze ruch o większej wartości. Sieć we wszystkich przypadkach preferuje poprawne zagrania.

W przypadku ruchów prowadzących do tego samego rezultatu punktowego z sekcji trzeciej *policy head* czasami preferuje jedno zagranie względem drugiego. W czwartym przykładzie można to uznać za zrozumiałe, ponieważ w zwykłych pozycjach na planszy najpierw należałoby zagrać ruch, który posiada kontynuację, a nie blokować analogiczny ruch przeciwnika. W warunkach testowych ruchy te prowadzą do tego samego rezultatu tylko dlatego, że są to jedyne dwie niewyjaśnione sytuacje na planszy. Z drugiej strony tak samo jest w przykładzie trzecim, a wskazania *policy head* dla obu ruchów są tam bardzo bliskie. Preferencje sieci dla różnych ruchów cechują się więc pewną niekonsekwencją. Ponadto, w przykładzie drugim ruchy są całkowicie równoważne, a sieć i tak wyraźnie preferuje jeden z nich.

Ciekawe jest, że *policy head* dobre wyniki uzyskała również w dwóch przykładach w sekcji poświęconej tedomari. Oznacza to, że już operacje przeprowadzane wewnątrz sieci neuronowej muszą dokonywać przewidywania przynajmniej dwóch ruchów naprzód. Możliwe, że sieć zawdzięcza tę cechę dodatkowemu wyjściu *policy head* przewidującemu następny ruch przeciwnika (punkt 4 w podrozdziale 3.4). W ostatnim przykładzie jest dostępnych więcej niż dwa ruchy i poprawny wybór zagrania jest zależny od parzystości liczby ruchów jednopunktowych bez kontynuacji. Sieć nie preferuje w tym przypadku poprawnego zagrania. Zamiast tego wszystkie ruchy wycenia w miarę podobnie.

W przykładach przedstawiających korytarze sieć nie posiada silnej preferencji odnośnie kolejności wchodzenia w nie. Co ciekawe, w każdym przykładzie minimalnie wyżej wycenia ruch zalecany przez teorię. Są to jednak różnice mniejsze niż odchylenia standardowe wycen tych ruchów.

We wszystkich badanych przykładach względne ułożenie badanych sytuacji nie miało większego wpływu na wyceny sieci. Można jednak zauważyć, że kiedy sytuacje były umieszczone daleko od siebie (w przeciwległych narożnikach) preferencje sieci odnośnie poprawnych zagrań są trochę bardziej wyraźne.

Podsumowując, *policy head* zauważa różnice w wartościach zagrań i proponuje większe ruchy. Ponadto, sieć potrafi przewidywać więcej niż jeden ruch naprzód, dzięki czemu *policy head* proponuje poprawne zagrania również w niektórych sytuacjach, kiedy ze względu na tedomari to ruch o mniejszej wartości jest tym poprawnym. Jednak w przypadku kilku ruchów o wartości zbliżonej do jednego punktu, wskazania sieci są bardzo zbliżone. Można więc powiedzieć, że *policy head* nie zna zasad dotyczących kolejności grania tych ruchów, które zostały przedstawione w podrozdziale 2.5. Być może dodatkowy trening, w którym za przykłady treningowe posłużyłyby pozycje z jednopunktowych końcówek, pomógłby sieci nauczyć się rozróżniać różne typy ruchów wartych około jednego punktu i preferować te spośród nich, które zaleca teoria.

5.3. Zapeszenie KataGo

Programy reimplementujące rozwiązanie użyte w AlphaZero, takie jak KataGo, Leela Zero czy ELF OpenGo, sugerują bardzo silne zagrania nawet bez użycia algorytmu przeszukiwania. Jednak ruchy te mogą być czasem słabe od strony taktycznej. Dla człowieka jest więc możliwe pokonać taki program, jeśli nie wykonuje on żadnych iteracji MCTS. Tymniemniej, już przy użyciu kilkuset iteracji MCTS program w większości partii pokonuje nawet najsilniejszych zawodowców.

Przeprowadzony test pokazał jednak, że da się ułożyć pozycję, w której ludzka wiedza wywiedziona z teorii matematycznej wystarczy do pokonania KataGo. W pozycji nazwanej problemem peszącym zawodowców KataGo nawet po dokonaniu miliona iteracji proponuje niepoprawny ruch. Proponowane zagranie posiada wartość mniejszą niż jeden punkt, mimo że na planszy jest dostępnych kilka ruchów jednopunktowych. Wykonanie w tej pozycji takiego ruchu jest błędem, po którym niemożliwe jest już wygranie partii.

Uzyskany rezultat nie powinien być zaskakujący, jeśli wziąć pod uwagę wyniki testów poświęconych wskazaniom *policy head*. Sieć KataGo nie rozróżnia zbyt precyzyjnie różnych rodzajów ruchów o wartości zbliżonej do jednego punktu. Ponadto, w problemie peszącym zawodowców dostępne są nie tylko zwykłe zagrania wchodzące w korytarze, ale także niepołączone łańcuchy mające dostęp do wielu korytarzy. To zagadnienie jest znacznie bardziej skomplikowane niż kwestia kolejności wchodzenia w korytarze. W problemie peszącym zawodowców dostępnych jest ponad dwadzieścia zagrań. Liczba ta nie musi maleć wraz z rozwojem partii, ponieważ po zagraniu ruchów wchodzących w korytarze, możliwe są dalsze ruchy kontynuujące to wchodzenie. Dlatego w najbliższych ruchach partii sieć będzie cały czas stawała przed wyborem przynajmniej kilkunastu opcji. Dokładne przeliczenie zaledwie sześciu ruchów wgłąb będzie więc wymagało ponad miliona operacji. Ponieważ sieć nie posiada silnych preferencji odnośnie ruchów w pojawiających się sytuacjach, to przeszukiwanie przebiega nieomalże na oślep i sieć nie jest w stanie znaleźć poprawnej sekwencji.

Problem peszący zawodowców jest sztuczną pozycją, która nigdy nie pojwiłaby się w prawdziwej partii. Interesujące byłoby przeprowadzenie dalszych badań, które pokazałyby, czy niedoskonałości KataGo ukazane przez ten problem prowadzą czasami w prawdziwych partiach do wyboru nieoptymalnych zagrań. Może się okazać, że czasami w jednopunktowych końcówkach występują pozycje na tyle skomplikowane, że KataGo może się pomylić i zakończyć grę z wynikiem o jeden punkt gorszym niż uzyskałby gracz posługujący się algorytmem wynikającym z kombinatorycznej teorii gier.

Jest również interesujące, czy byłoby możliwe przeprowadzenie dodatkowego treningu używającego pozycji z problemu peszącego zawodowców, który pozwoliłby programowi zrozumieć wnioski płynące z teorii matematycznej. Autor programu KataGo dokonał już podobnego eksperymentu dla zagadnienia tak zwanych *tsumego* (problemów życia i śmierci). W artykule *Deep-Learning the Hardest Go Problem in the World* [20] opisał dodatkowy trening KataGo bazujący na pozycjach z jednego problemu życia i śmierci, dzięki któremu KataGo było w stanie odnaleźć w tym problemie wiele silnych zagrań, nawet takich, których nie odkryli goiści studiujący wcześniej ten problem. Może się jednak okazać, że kombinatoryczna teoria gier opisująca końcową fazę gry Go na tyle różni się od umiejętności potrzebnych do zrozumienia pozostałych aspektów gry, że rozwiązania użyte w KataGo nie wystarczą do jej dogłębnego zrozumienia. Taki rezultat pokazywałby, że algorytmy bazujące na podejściu zero (albo semi-zero) posiadają pewne ograniczenia w odniesieniu do Go.

Słownik terminów

atari Sytuacja, w której łańcuch kamieni posiada tylko jeden oddech. Jeśli w takiej sytuacji przeciwnik zabierze ten oddech (postawi kamień na przecięciu stanowiącym ten oddech), to zbije dany łańcuch. Także: ruch doprowadzający do takiej sytuacji. Patrz też: postawić pod atari

yose Końcowa faza partii. Za początek yose uznaje się moment, w którym wyjaśniony zostaje status wszystkich łańcuchów kamieni i mniej więcej znane są granice terytoriów. W yose pozycję można analizować jako sumę teoriomnogościową lokalnych sytuacji w różnych częściach planszy

drabinka Podstawowa technika łapania kamieni polegająca na nieustannym stawianiu pod atari łańcucha przeciwnika. Za każdym razem kiedy przeciwnik wydłuża swój łańcuch, pozostają mu tylko dwa oddechy, dzięki czemu można znowu zagrać atari. Drabinka została przedstawiona w tekście na diagramie 3.1

groźba do ko Ruch zagrażający uzyskaniem dużego zysku wykonywany zaraz po tym, jak przeciwnik zbił ko. Jeśli przeciwnik odpowie na groźbę, to gracz będzie mógł odbić ko (ponieważ w tej chwili sytuacja na planszy już będzie się różnić od sytuacji przed poprzednim zbiciem ko). Patrz też: walka ko

ko Sytuacja, w której kamień jednego z graczy stoi pod atari, jednak jeśli przeciwnik go zbije (zbije ko), to jego dopiero co postawiony kamień, będzie stał pod atari. W takiej sytuacji reguły Go zabraniają graczowi zbić ten dopiero co postawiony kamień (odbić ko), ponieważ prowadziłoby to do powtórzenia sytuacji na planszy. Patrz też: walka ko

komi Ustalona liczba punktów dodawana do końcowego wyniku Białych służąca wyrównaniu początkowej przewagi Czarnych, jaką daje im fakt wykonywania pierwszego ruchu w partii. W zasadach japońskich komi jest równe $6,5$ punktu. Połówka punktu w komi zapewnia, że partia nie zakończy się remisem

kontynuacja Drugi ruch w lokalnej sekwencji wykonywany przez tego samego gracza, co pierwszy ruch. Gracz może zagrać kontynuację, jeżli przeciwnik nie odpowiedział na jego pierwszy ruch

kształt Lokalne ułożenie kamieni. Pojęcie często używane w odniesieniu do kamieni tego samego koloru, które są połączone lub mogą zostać połączone w łatwy sposób

łańcuch Zbiór połączonych kamieni. Łańcuch pozostaje na planszy, dopóki posiada przynajmniej jeden oddech

łączyć (kamienie/łańcuchy) Zagrać ruch na przecięciu sąsiadującym z dwoma różnymi łańcuchami danego gracza i tym samym uczynić z nich jeden łańcuch

martwy (kamień/łańcuch) Taki łańcuch, którego nie da się uratować przed zbiciem. Nawet jeśli gracz próbowałby uratować taki łańcuch, to jeśli przeciwnik będzie poprawnie odpowiadał, to zdoła go zbić. Po zakończeniu partii podczas liczenia punktów martwe łańcuchy są zdejmowane z planszy i traktowane tak samo jak zbite kamienie

oddech (kamienia/łańcucha) Niezajęte przecięcie sąsiadujące z kamieniem należącym do danego łańcucha

odpowiedź Drugi ruch w lokalnej sekwencji wykonywany przez przeciwnika gracza, który wykonywał pierwszy ruch

oko Niezajęte przecięcie otoczone przez łańcuch jednego z graczy albo przez kamienie jednego gracza należące do różnych łańcuchów, o ile przeciwnik nie ma możliwości zabrania wszystkich pozostałych oddechów jednemu z tych łańcuchów i postawienia go pod atari (jeśli przeciwnik ma taką możliwość, to przecięcie nazywa się fałszywym okiem)

połączone kamienie Kamienie tego samego koloru należące do jednego łańcucha. Kamienie tego samego koloru K_p i K_o są połączone wtedy i tylko wtedy, gdy istnieje ciąg kamieni tego samego koloru $K_1, K_2, \ldots, K_n$ taki, że $K_1 = K_p$, $K_n = K_o$ oraz dla $1 \leq i \leq n-1$ kamienie K_i i K_{i+1} stoją na sasiądujących przecięciach. Pojęcie używane też w drugim, mniej formalnym znaczeniu: kamienie tego samego koloru, których nie da się rozciąć. Nawet jeśli przeciwnik podejmie próbę ich rozdzielenia, to jeśli gracz będzie poprawnie odpowiadał, to zdoła połączyć je w jeden łańcuch

postawić pod atari Zagrać ruch, który zabiera przedostatni oddech łańcuchowi przeciwnika (a zatem grozi zbiciem w następnym ruchu)

samobójczy ruch Ruch, który prowadziłby do odebrania ostatniego oddechu łańcuchowi grającego. Ruchy samobójcze są niedozwolone w większości zestawów reguł

sąsiadujące przecięcia Przecięcia, których współrzędne różnią się na dokładnie jednej pozycji o dokładnie jeden

sente Inicjatywa. Także: ruch, na który przeciwnik powinien odpowiedzieć, ruch zachowujący inicjatywę. W teorii końcówek pojęcie używane w znaczeniu lokalnym: ruch podwyższający lokalną temperaturę, ruch, dla którego wartość kontynuacji przekracza pewny zysk

walka ko Sytuacja, w której na planszy jest obecne ko i każdemu z graczy zależy na tym, żeby je wygrać, to znaczy, żeby zbić ko i następnie nie pozwolić przeciwnikowi na jego odbicie, na przykład poprzez połączenie ko, czyli zagranie ruchu na przecięciu stanowiącym jedyny oddech kamienia, przy pomocy którego gracz zbił ko. W takiej sytuacji kiedy jeden z graczy zbija ko, przeciwnik gra groźbę do ko, próbując zmusić gracza do zagrania w innej części planszy. Jeśli pierwszy gracz odpowie na groźbę, przeciwnik odbija ko i tym razem to pierwszy gracz szuka groźby do ko

wydłużyć (kamień/łańcuch) Zagrać ruch na przecięciu, które sąsiaduje z jednym kamieniem należącycm do łańcucha danego gracza

wymiana Lokalna sekwencja złożona z ruchu jednego gracza i odpowiedzi jego przeciwnika

zabić (kamień/łańcuch) Uczynić łańcuch przeciwnika martwym, zazwyczaj poprzez uniemożliwienie mu stworzenia dwojga oczu. Porównaj: złapać

zbić (kamień/łańcuch) Zabrać ostatni oddech łańcuchowi przeciwnika. Kiedy gracz zbija kamienie przeciwnika, zdejmuje je z planszy

złapać (kamień/łańcuch) Uczynić łańcuch przeciwnika martwym, zazwyczaj poprzez otoczenie go. Pojęcia tego używa się głównie w odniesieniu do łańcuchów, które widać, że nie mają szans na stworzenie dwojga oczu. Porównaj: zabić

żywy (kamień/łańcuch) Taki łańcuch, którego nie da się zbić. Nawet jeśli przeciwnik próbowałby mu zabrać oddechy, to jeśli gracz będzie poprawnie odpowiadał, to przeciwnikowi nie uda się go zbić. Przykładem żywych łańcuchów są łańcuchy posiadające dwoje oczu

Bibliografia

[1] Victor Allis. "Searching for solutions in games and artificial intelligence". English. ISBN: 9789090074887 OCLC: 905509528. Prac. dokt. S.l.: s.n.], 1994.

[2] Elwyn Berlekamp. "The economist's view of combinatorial games". W: *Games of No Chance: Combinatorial Games at MSRI*. University Press, 1996, s. 365–405.

[3] Elwyn R. Berlekamp, John Horton Conway i Richard K. Guy. *Winning ways for your mathematical plays*. 2nd ed. Natick, Mass: A.K. Peters, 2001. ISBN: 978-1-56881-130-7 978-1-56881-142-0 978-1-56881-143-7 978-1-56881-144-4.

[4] Elwyn R. Berlekamp i David Wolfe. *Mathematical go: chilling gets the last point*. eng. Nachdr. OCLC: 246242023. Wellesley, Mass: Peters, 1997. ISBN: 978-0-923891-36-7 978-1-56881-032-4.

[5] B. Bouzy i B. Helmstetter. "Monte-Carlo Go Developments". W: *Advances in Computer Games*. Red. H. Jaap Herik, Hiroyuki Iida i Ernst A. Heinz. Boston, MA: Springer US, 2004, s. 159–174. ISBN: 978-1-4757-4424-8 978-0-387-35706-5. DOI: 10.1007/978-0-387-35706-5_11. URL: http://link.springer.com/10.1007/978-0-387-35706-5_11 (term. wiz. 04. 09. 2020).

[6] Bernd Brügmann. *Monte Carlo Go*. 1993.

[7] John Horton Conway. *On numbers and games*. 2nd ed. Natick, Mass: A.K. Peters, 2001. ISBN: 978-1-56881-127-7.

[8] Markus Enzenberger. *The Integration of A Priori Knowledge into a Go Playing Neural Network*. Spraw. tech. 1996.

[9] David Fotland. *Knowledge Representation in The Many Faces of Go*. URL: http://www.smart-games.com/knowpap.txt (term. wiz. 04. 09. 2020).

[10] Chris J. Maddison i in. "Move Evaluation in Go Using Deep Convolutional Neural Networks". W: *arXiv:1412.6564 [cs]* (kw. 2015). arXiv: 1412.6564. URL: http://arxiv.org/abs/1412.6564 (term. wiz. 04. 09. 2020).

[11] M. Müller. "Computer go as a sum of local games: an application of combinatorial game theory". W: 1995. DOI: 10.3929/ethz-a-001440588.

[12] Gian-Carlo Pascutto. *Leela Zero*. en. URL: https://zero.sjeng.org/home (term. wiz. 04. 09. 2020).

[13] David Silver i in. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". W: *arXiv:1712.01815 [cs]* (grud. 2017). arXiv: 1712.01815. URL: http://arxiv.org/abs/1712.01815 (term. wiz. 04. 09. 2020).

[14] David Silver i in. “Mastering the game of Go without human knowledge”. en. W: *Nature* 550.7676 (paź. 2017), s. 354–359. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature24270. URL: http://www.nature.com/articles/nature24270 (term. wiz. 04. 09. 2020).

[15] T. Takizawa. “An Application of Mathematical Game Theory to Go Endgames: some Width-Two-Entrance Rooms With and Without Kos”. W: 2002.

[16] Yuandong Tian i Yan Zhu. “Better Computer Go Player with Neural Network and Long-term Prediction”. W: *arXiv:1511.06410 [cs]* (lut. 2016). arXiv: 1511.06410. URL: http://arxiv.org/abs/1511.06410 (term. wiz. 04. 09. 2020).

[17] Antti Törmänen. *Rational Endgame*. English. OCLC: 1096284936. Hebsacker Verlag, 2019. ISBN: 978-3-937499-09-3 978-3-937499-10-9.

[18] David Wolfe. “Go endgames are PSPACE-hard”. W: *More Games of No Chance*. Cambridge University Press, 2001, s. 125–136.

[19] David J. Wu. “Accelerating Self-Play Learning in Go”. W: *arXiv:1902.10565 [cs, stat]* (lut. 2020). arXiv: 1902.10565. URL: http://arxiv.org/abs/1902.10565 (term. wiz. 04. 09. 2020).

[20] David J. Wu. *Deep-Learning the Hardest Go Problem in the World*. Grud. 2019. URL: https://blog.janestreet.com/deep-learning-the-hardest-go-problem-in-the-world/ (term. wiz. 11. 11. 2020).

[21] David J. Wu. *lightvector/KataGo*. original-date: 2019-02-26T04:57:14Z. Wrz. 2020. URL: https://github.com/lightvector/KataGo (term. wiz. 10. 09. 2020).

Dodatek A

Strategia gorąca

W ekonomicznym Go najprostsza strategia polega na wybieraniu za każdym razem ruchu o najwyższej wartości, o ile tylko wartość ta przekracza wysokość podatku. Berlekamp w artykule *The Economist's View of Combinatorial Games* [2] nazywa ją strategią gorącą i pokazuje, że jest ona naiwna. Przedstawia przykład gry, w której zastosowanie tej strategii prowadzi do niekorzystnego rezultatu.

Berlekamp zapisuje swój przykład w notacji kombinatorycznej teorii gier jako sumę $\boldsymbol{A} + \boldsymbol{B}$, gdzie

$$\boldsymbol{A} = 1||||10||0| - 20||| - 21, \ \boldsymbol{B} = 1|||0|| - 18$$

Diagram A.1 stanowi ilustrację takiej gry. Jeżeli Czytelnik poznał grę Go dopiero w niniejszej pracy, to przedstawiona pozycja może wyglądać na bardzo skomplikowaną. Notacja kombinatorycznej teorii gier może być wówczas łatwiejsza do zrozumienia. Z drugiej strony, dla gracza Go analiza diagramu powinna być znacząco prostsza.

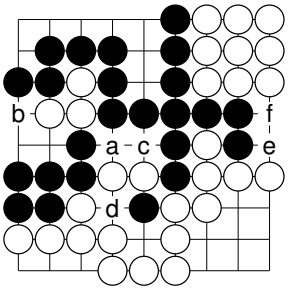


Diagram A.1: Przykład z artykułu Berlekampa Podatek = 1

Przyjmujemy, że lokalne wyniki w sytuacji w prawym górnym narożniku oraz w sytuacji obejmującej zarówno lewą bandę, jak i środek planszy, są równe 0. Możemy tak przyjąć na mocy stwierdzenia 2.1.3.

Wymiana *e*-*f* to przywilej Białych, brak odpowiedzi Czarnych prowadziłby do lokalnego wyniku -18. Wymiana *a*-*b* to również przywilej Białych. Gdyby Czarne nie odpowiedziały w *b*, Białe mogłyby dać atari na łańcuch sześciu czarnych kamieni i w następnym ruchu go zbić. Jednakże Czarne, zanim odpowiedzą na ruch Białych mogą dać atari na trzy kamienie Białego w *c*. Dopiero kiedy Białe połączą się w *d*, Czarne dograją w *b*.

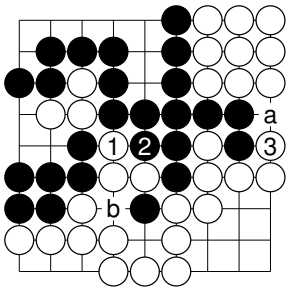


Diagram A.2: Nietypowa kolejność ruchów Białych

Berlekamp przedstawia przebieg partii, w której Białe zastawiają pułapkę na przeciwnika używającego strategii gorącej. Po tym, jak Czarne dają atari na trzy kamienie ruchem ❷, Białe grożą, że uratują dziewięć kamieni w prawym górnym narożniku. W tej sytuacji wartości ruchów *a* i *b* są równe odpowiednio 9 i 10 punktów. Czarne, kierując się strategią gorącą, wybierają zbicie trzech kamieni. Białe kończą partię ruchem w *a*.

Czarne wpadły w pułapkę. Gdyby odpowiedziały na groźbę Białych, zakończyłyby z wynikiem lepszym o osiem punktów. Przy zastosowaniu przez Czarne strategii gorącej Białe wykonują ostatni ruch (zdobywają tedomari). Wprawdzie oznacza to, że muszą zapłacić podatek o jeden raz więcej, jednak nie jest to duży kłopot.

Wysokość podatku jest równa 1, dlatego nie jest on w stanie zniwelować straty poniesionej przez Czarne.

W przypadku zastosowania strategii książkowej opisanej w podrozdziale 2.3 Czarne nie wpadłyby w pułapkę. Po ruchu Białych grożących uratowaniem dziewięciu kamieni lokalna temperatura wzrosła powyżej wysokości podatku. Dlatego Czarne odpowiedziałyby na ten ruch, zamiast szukać ruchu o najwyższej wartości na planszy. Berlekamp dowodzi, że przy optymalnej grze przeciwnika zagrania wybrane zgodnie ze strategią książkową również są optymalne i partia kończy się remisem. Przy dowolnym innym sposobie gry przeciwnika jest to strategia nieprzegrywająca. Oznacza to, że gracz posługujący się strategią książkową być może nie wykorzysta pewnych błędów przeciwnika i przegapi szansę na odniesienie zwycięstwa. Jednakże może mieć pewność, że w każdej partii w najgorszym przypadku osiągnie remis.

Dodatek B

Partia ELF OpenGo

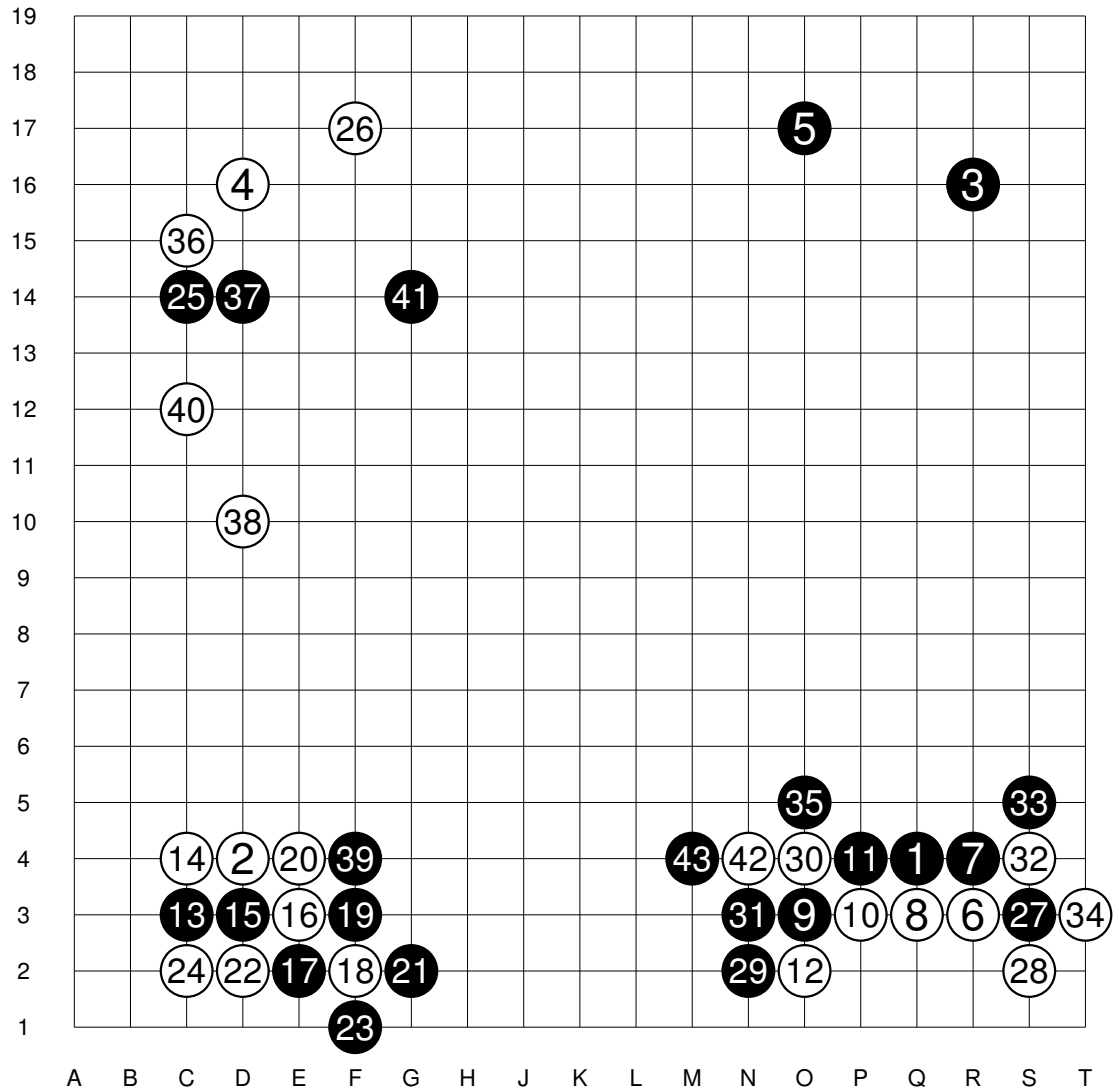


Diagram B.1: Oficjalna partia, którą program ELF OpenGo rozegrał z jednym z czołowych koreańskich profesjonalistów. Program grał Białymi. Ruch (42) jest próbą ucieczki z drabinki. Ponieważ na drodze drabinki nie stoi kamień Białych, Czarne chwytają dwa Białe kamienie ruchem ●43. Ruch Białych jest prostym błędem, którego uczą się unikać już gracze początkujący.

Dodatek C

Pozycja testowa dla mapy przynależności

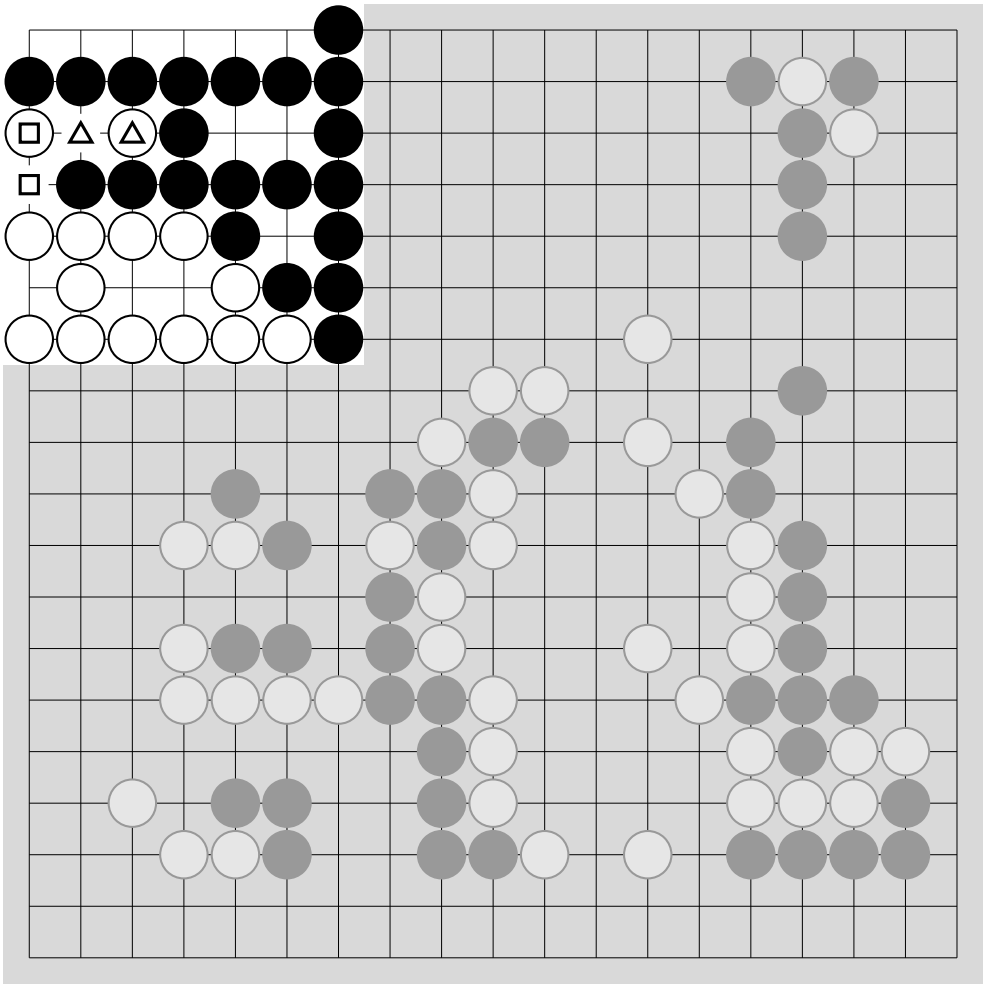


Diagram C.1: Jedna z tysiąca całoplanszowych pozycji przedstawionych KataGo w ramach pierwszego przykładu testowego w sekcji poświęconej ruchom z kontynuacją. Test jest wykonywany dla pozycji z osiemdziesiątego ruchu partii (której wszystkie ruchy były zagrane w wyszarzonej części planszy). Pozycja testowa ustawiona jest w lewym górnym narożniku.

Dodatek D

Pozycje testowe dla *policy head*

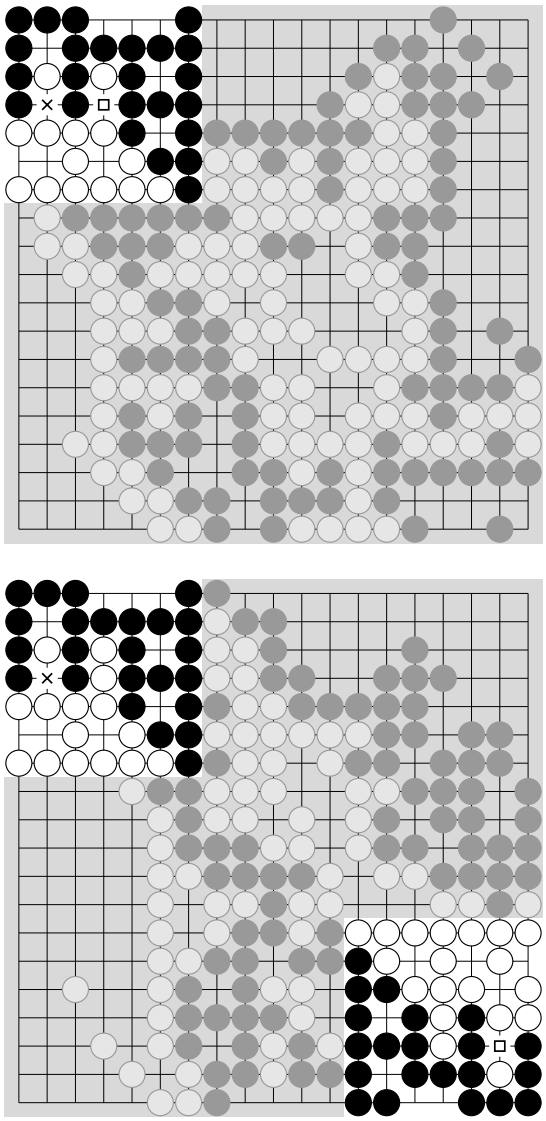


Diagram D.1: Pierwszy przykład z sekcji poświęconej ruchom o różnych wartościach. Test jest przeprowadzany dla pozycji końcowej każdej z tysiąca wygenerowanych partii. Na górze wersja testu z sytuacjami umieszczonymi w tym samym narożniku. Na dole wersja testu, w której te same dwie lokalne sytuacje zostały umieszczone w przeciwległych narożnikach.

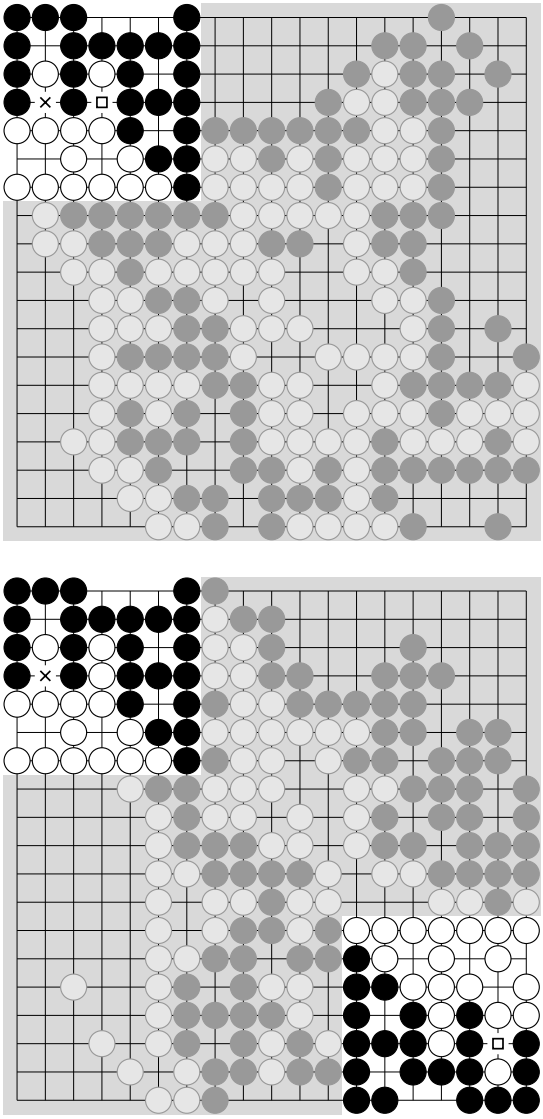


Diagram D.2: Pierwszy przykład z sekcji poświęconej korytarzom. Wersja testu z sytuacjami umieszczonymi w przeciwległych narożnikach została skonstruowana poprzez umieszczenie pozycji z lewego górnego narożnika z zamienionymi kolorami w prawym dolnym narożniku. W pozycji w prawym dolnym narożniku jest również dodany jeden kamień Białych wchodzący w jeden z korytarzy Czarnych.