Optymalizacja, projekt 2 - Raport

Celem projektu było stworzenie programu znajdującego optymalną teoriogrowo strategię do gry Liar's Dice dla 3 graczy, projekt naturalnie dzielił się na 2 części - gra dwuosobowa oraz gra trzyosobowa.

1 Gra dwuosobowa

Dla gry dwuosobowej zaimplementowane została metoda Kollera przedstawiona przez Marca Lanctota w publikacjach dołączonych do treści projektu.

Unika ona operowania na macierzy gry w postaci normalnej (która posiada w tym przypadku rozmiary uniemożliwiające taką realizację), a zamiast tego operuje na sekwencjach i tzw. information setach. Wiersze i kolumny tak jak w postaci normalnej odpowiadają graczom, ale zamiast czystych strategii trzymane są tam wczesniej wspomniane sekwencje ruchów, i w macierzy element o współrzędnych (a_1, a_2) reprezentuje wypłatę w momencie wykonania ruchów a_1 przez pierwszego gracza i a_2 przez drugiego gracza. Kiedy wykonanie takich ruchów nie jest jeszcze wypłatą, albo nie jest legalną kombinacją, macierz zawiera 0.

W publikacji dla zwięzłości zapisu użyto liter do oznaczania betów, będę podążał za tym zapisem, to znaczy:
$X1abc$ - oznacza wyrzucenie przez gracza X jedynki oraz zrobienie beta a (1, 1), na co gracz Y odpowiedział betem b(1,2) a gracz X betem c(1,3) (lub (2, 1)) W tak rozumianej grze możemy reprezentować macierz wypłat, jednak musimy zmienić constrainty narzucane przez program liniowy, tak jak w normalnej wersji wystarczyło sumowanie się strategii do 1 tutaj to nie ma sensu. Ponieważ zmienne reprezentują strukturę drzewiastą można to wykorzystać i oznaczyć, że wykonanie pustego ruchu ma prawdopodobieństwo 1, a wszystkie ruchy w danym information secie sumują się do ruchu poprzedniego.

Na takiej macierzy używamy tricku podanego w książce Matouska do znajdywania equilibrium w grach o sumie zerowej mając na uwadze to, że constrainty są bardziej złożone, ponieważ jednak tylko w jednym z nich występuje niezerowy stały element (root drzewa), nadal będziemy minimalizować (bądź maksymalizować) jeden element, jednakże constrainty narzucone przez transpozycję początkowej macierzy constraintów nadal będą obowiązywały.

Rozwiązanie zostało sprawdzone dla danych takich samych na jakich operował Marc Lanctot (różnica taka, że mydłem jest największa liczba, a nie 1) i dla liczby oczek równej 2 3 oraz 4 i wyniki się pokrywały.

Dodatkowo w celu rozeznania się z metodą Kollera wymyślona została prostsza gra i rozwiązana - polegająca na tym, że obaj gracze rzucają kostką, jeden gracz zgaduje sumę oczek, a drugi mówi czy liczba jest większa czy mniejsza. Implementacja również zawiera się w kodach źródłowych dołączonych.

2 gra trzyosobowa

rozwiązanie dla trzech osób (i formalnie dla n-osób, chociaż dla n ¿ 3 wymagałoby to koszmarnie wielkiej mocy obliczeniowej) korzysta na intuicyjnym poziomie z dwóch następujących faktów:

- rozwiązanie optymalne musi być przygotowane na każdą odpowiedź zagraną przez przeciwników
- gdyby przeciwnicy połączyli siły to nie graliby gorzej niż osobno.

więc szukając strategii gracza x w grze trzyosobowej przeciwko graczom y i z, szuka się strategii optymalnej gracza x w grze dwuosobowej odpowiadającej grze w której gracze y i z porozumiewają się ze sobą.

W Liar's dice działa to tak, że gracz X tak samo jak poprzednio wykonuje jeden bet (np: $X1abc \rightarrow X1abce$, zaś gracz Y, który odpowiada kooperującym pozostałym dwóm graczom ma informacje o dwóch kostkach oraz przy każdym ruchu ma dwie możliwości:

- call jako pierwszy gracz ($Y11abc \rightarrow Y11abci$, gdzie i reprezentuje call
- zwykly bet jako pierwszy gracz i jakiś bet jako drugi gracz

Gdzie jeżeli któryś z dwóch graczy z zespołu dwuosobowego odpada to jest to w macierzy wypłat reprezentowane przez stratę dla zespołu. Jak już wspominałem jest to szalenie kosztowne obliczeniowo, o ile dla kostek z 2 możliwościami (coin-throws) rozwiązanie gry 3-osobowej trwa na sage'u jednocyfrową liczbę sekund, to dla kostek z 3 możliwościami rozwiązanie gry 3-osobowej trwa już ponad godzinę.

możliwe optymalizacje:

- powyższa struktura gry trzyosobowej narzuca optymalizację zmniejszającą ilość zmiennych - zamiast trzymać informację o dwóch betach dwóch graczy z zespołu, w przypadku gdy druga opcja została wybrana można trzymać informację o drugim becie (pierwszy nie wnosi żadnej informacji). Znacząco to redukuje ilość (pierwiastkuje) zmiennych, gdyż nie trzymamy nic nie wnoszących zmiennych, przykład:
  Y11abe, Y11ace, Y11ade - wszystkie reprezentują ten sam ruch.
- ponieważ liczy się tylko ilość kostek, można trzymać tylko niemalejące ciągi wyrzuconych kostek z zespołu, ta optymalizacja ma mniejsze znaczenie dla trzech graczy niż przy większych gracz, ponieważ usuwa jedynie zmienne Y21... , jednak dla większej ilości graczy mialaby ogromne znaczenie.