Inverse Kinematics for Binary Trees

Marcin Januszkiewicz, Jakub Kowalski

24 stycznia 2012

Spis treści

1 Sformułowanie zadania

Zagadnienie kinematyki odwrotnej (*ang.* inverse kinematics) dotyczy reprezentacji kończyn (ogólnie tzw. efektora) za pomocą geometrycznego modelu w którym kości estymowane są przez linie, natomiast stawom odpowiadają kąty. Zadaniem kinematyki odwrotnej jest znalezienie takich położeń stawów, dla których osiągana jest pożądana pozycja kończyny, np. zamocowana w pewnym punkcie startowym dosięga ona zadanego punktu końcowego. Możliwe położenia stawów określone są przez minimalny i maksymalny kąt rozwarcia. Chromosom jest to lista kątów odpowiadających rozwarciom kolejnych stawów kończyny.

Rozpatrywana przez nas rozszerzona wersja standardowego zagadnienia dotyczy „kończyn" o strukturze pełnego drzewa binarnego. Zadanie polega na znalezieniu ustawienia drzewa w którym każdy punkt końcowy dotyka określonego punktu docelowego (punktów docelowych jest tyle samo co liści drzewa) a krawędzie drzewa nie stykają się z żadną znajdującą się na planszy przeszkodą.

1.1 Format pliku wejściowego

Scenariusz (plik zawierający informację o planszy oraz właściwościach drzewa) jest plikiem o następującej konstrukcji (linie rozpoczynające się od # traktowane są jako komentarz):

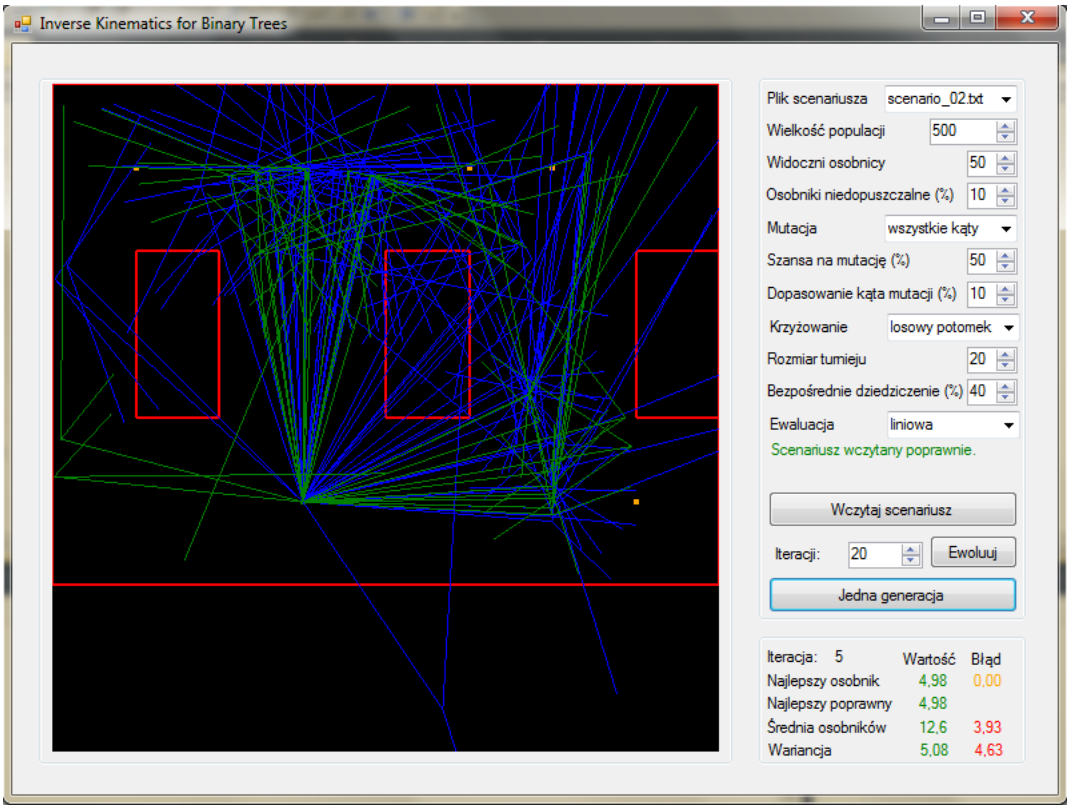
```
# Wielkość planszy w poziomie
8
# Wielkość planszy w pionie
6
# Punkt startowy (wszystkie punkty podane są w notacji X Y)
3 5
# Punkty docelowe
1 1
5 1
6 1
7 5
# Specyfikacja drzewa
# Początkowy ciąg znaków (R|L)+ koduje umiejscowienie danej krawędzi w drzewie
# Każda krawędź opisana jest za pomocą długości oraz minimalnej
# i maksymalnej miary kąta stawu względem krawędzi poprzedzającej
L 4 0 360
LL 2 0 360
LR 2 0 360
R 3 0 360
RL 4 0 360
RR 1 0 360
# Znajdujące się na planszy przeszkody definiowane są jako linie.
# Linia pomiędzy punktami (x1, y1) a (x2, y2) podawana jest w formacie
# x1 y1 x2 y2
1 2 2 2
2 2 2 4
2 4 1 4
1 4 1 2
```

```
4 2 5 2
5 2 5 4
5 4 4 4
4 4 4 2
7 2 8 2
8 4 7 4
7 4 7 2
```

Przypisanie punktów docelowych do liści drzewa odpowiada trawersowaniu go od lewej do prawej, tak więc pierwszy punkt docelowy zostanie przypisany do krawędzi `LL`, drugi do `LR`, trzeci do `RL` itd.

2 Obsługa programu

Graficzny interfejs użytkownika, wykorzystywany przez program testowy, przedstawiony jest na rysunku 1.



Rysunek 1: Interfejs programu

Program pozwala wybrać jeden z przygotowanych plików scenariusza oraz określić podstawowe parametry ewolucji a także wybrać kombinację stosowanych przy ewolucji operatorów. Po wczytaniu scenariusza można uruchomić ewolucję od razu na kilka pokoleń zgodnie z przekazanymi danymi lub sprawdzać jej zachowanie krok po kroku.

Statystyki aktualnej populacji pozwalają ocenić osobników (pokazywane są wartości błędu i minimalizowanej funkcji celu dla osobnika najlepszego w całej populacji, najlepszego bezbłędnego oraz wartości średnie), a także ocenić zbieżność populacji poprzez zmieniające się wartości wariancji (dla błędu oraz funkcji oceny) i oryginalności populacji.

Wizualizacja przedstawia planszę zakodowaną w scenariuszu. Czerwone linie oznaczają przeszkody, zielony punkt jest punktem startowym, natomiast pomarańczowe punkty są punktami docelowymi. Na wizualizacji zaznaczeni są również najlepsi osobnicy w liczbie zdefiniowanej przez uzytkownika. Niebieskie linie pokazują przebieg osobników błędnych, natomiast zielone bezbłędnych.

3 Ewolucja

Zastosowany algorytm ewolucyjny częściowo bazuje na algorytmie IDEA [1], jednak ze względu na specyfikę problemu zastosowane zostały rozwiązania dedykowane. Drzewiasta struktura kończyny pozwala na (przynajmniej w pewnym zakresie) niezależne rozpatrywanie ewolucji niektórych poddrzew, co wpłynęło na sposób konstruowania operatorów ewolucyjnych.waniu.

3.1 Heurystyki

Zakodowane w operatorach heurystyki starają się określić które poddrzewa warto w danej iteracji mutować oraz krzyżować. W momencie gdy zdecydujemy, że do pewnej głębokości krawędzie są w położeniu odpowiednim, to wystarczy manipulować krawędziami z niższych poziomów, im większy numer iteracji tym precyzyjniej. Wybór poddrzew podlegających krzyżowaniu również może stanowić pole do działania dla heurystyk, choć wszystkie stosowane przez nas operatory korzystają tylko z podstawowego podziału na pierwszym poziomie rozgałęzienia.

3.2 Operatory ewolucyjne

Aplikacja pozwala na wybór operatorów z puli kilku przygotowanych.

3.2.1 Krzyżowanie

Przygotowane zostały dwa operatory krzyżowania o wspólnym schemacie działania. Operatory przyjmują jako argument listę ścieżek określającą poddrzewa które mają zostać skrzyżowane. Z puli wszystkich chromosomów wybierana jest losowo próbka o wielkości zadanej parametrem *rozmiar turnieju*. Następnie dla każdego poddrzewa, z próbki rodziców tworzone jest nowe poddrzewo które zostaje wstawione do chromosomu wynikowego. Po wykonaniu tej operacji dla wszystkich poddrzewa, wygenerowany zostaje jeden potomek.

Oba operatory rozróżnia właśnie sposób tworzenia poddrzewa z próbki rodziców. Pierwszy algorytm (*losowy potomek*) z prawdopodobieństwem zadanym parametrem *bezpośrednie dziedziczenie* otrzymuje poddrzewo najlepszego (na tym poddrzewie) ze swoich rodziców. W przeciwnym wypadku wybieranych jest dwóch rodziców: najlepszy na danym poddrzewie oraz losowy z próbki i dla pewnego losowego zaburzenia $\beta \in [0, 1]$, ma 50% na ewolucję w potomka c_1 i 50% na ewolucję w c_2, zgodnie ze wzorami:

$$c_1 = \frac{\alpha_1 + \alpha_2 + \beta(\alpha_1 - \alpha_2)}{2} \qquad c_2 = \frac{\alpha_1 + \alpha_2 + \beta(\alpha_2 - \alpha_1)}{2}$$

Drugi algorytm, *najlepszy potomek*, nie ma możliwości bezpośredniego dziedziczenia poddrzewa, ewaluuje jednak obie wersje potomków (c_1 i c_2) i wybiera lepszego z nich.

3.2.2 Mutacja

Mutacja jest w naszym algorytmie bardzo ważnym operatorem, ponieważ odpowiednio sparametryzowana pozwala na poszukiwanie miar kątów w poddrzewach, które mogą tego wymagać. Przygotowane zostały trzy operatory mutacji. Pierwszy, zapożyczony z algorytmu IDEA polega po prostu na losowym zaburzaniu (o dowolna wartość) każdego kąta z pewnym prawdopodobieństwem zadanym parametrem *szansa na mutację*. Mutacja ta nie jest w żaden sposób precyzyjna i ustawienie dużej szansy na jej zajście, zaburza zbieżność algorytmu.

Druga wersja mutacji (*jedna głębokość*) określa na podstawie numer u iteracji algorytmu poziom drzewa, który należy mutować (z pewnym losowym wahaniem ± 1). Następnie każde poddrzewo na tej głębokości obraca (z szansą *szansa na mutację*) o pewną wartość z zakresu $[-\alpha, \alpha]$, gdzie α jest liczbą stopni podaną w parametrze *dopasowanie kąta mutacji*. Mutacja ta nie dotyka poddrzew o optymalnej wartości funkcji celu.

Trzecia mutacja, *losowe poddrzewo*, **TODO – TOUPDATE**

3.3 Funkcja celu

Podstawą wartości funkcji jest dystans od liścia do przypisanego mu punktu docelowego, wyznaczany jako odległość w linii prostej. Istnieje możliwosc wybrania funkcji ze względu na to czy czynnik ten jest brany pod uwagę liniowo czy też kwadratowo.

Oprócz odległości do celu wyznaczana jest również miara niepoprawności osobnika, liczona jako liczba przecięć segmentów chromosomu z przeszkodami. Docelowo algorytm dąży do trzymania jedynie niewielkiej liczby osobników niepoprawnych, rokujących na uzyskanie dobrych potomków.

Ocena węzła drzewa jest sumą ocen jego poddrzew.

4 Testy

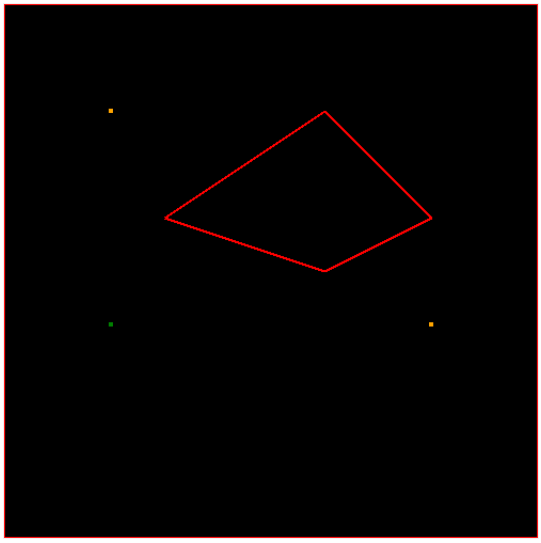
Algorytm był testowany na kilku przygotowanych scenariuszy i dla różnych wartości parametrów.

4.1 Scenariusze testowe

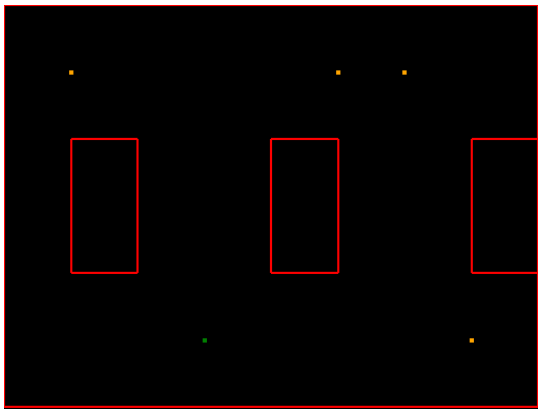
TODO – TOUPDATE Stworzonych zostało pięć scenariuszy testowych, które zostały przedstawione na rysunkach 2, 3, 4, 5, 6.

4.2 Wyniki

TODO – TOUPDATE Osiągane wyniki nie są zadowalające ponieważ algorytm, pomimo sprawnego umiejscowienia nadgarstka i wyewoluowania osobników znajdujących się blisko celu, ma problem z osiągnięciem optymalnego rozwiązania. Dla nawet małej populacji (50 osobników) w scenariuszach testowych nadgarstek był obiecująco umieszczany w przeciągu pierwszych 10 – 20 iteracji. Problem ze zbieżnością całego chromosomu wynika zapewne ze



Rysunek 2: Scenariusz testowy 1



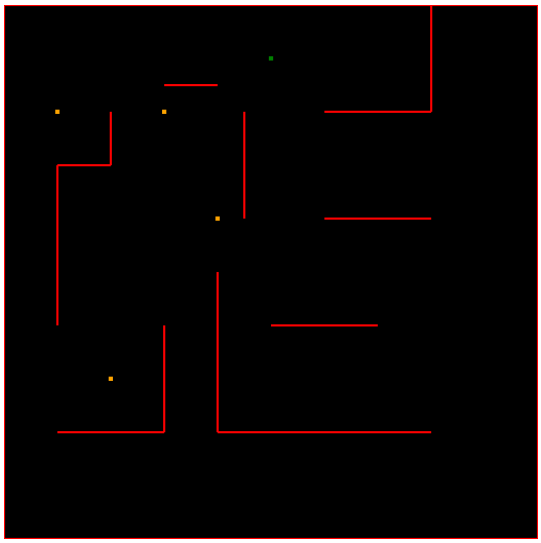
Rysunek 3: Scenariusz testowy 2

specyfiki zadania, które w szczególności nie określa odgórnie jaki palec ma zmierzać do jakiego celu. Powoduje to, że krzyżowanie osobników obierających różne dopasowania oddala populację od osiągnięcia rozwiązania.

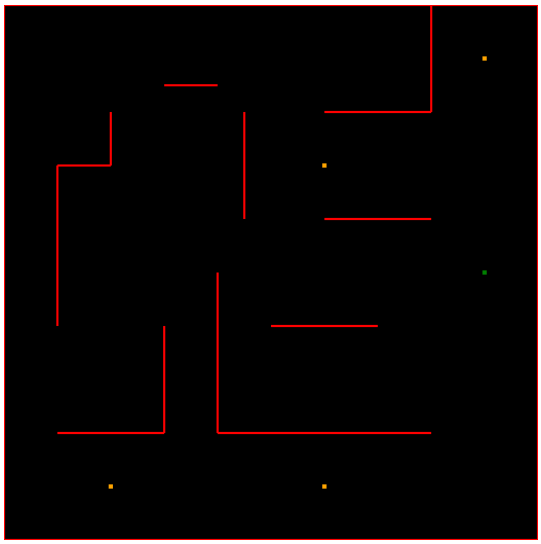
5 Podsumowanie

TODO – TOUPDATE Podsumowując, rozszerzenie dziedziny zadania poskutkowało znacznym jego utrudnieniem. Różnicę tę widać na przykładzie szybkości osiągania połowicznego celu jakim jest umiejscowienie nadgarstka. Aby podobne rezultaty osiągać dla całego osobnika, należałoby rozbudować chromosom przechowując w nim informację o najbardziej obiecującym przypasowaniu palców do punktów docelowych. Wiązałoby się to również z modyfikacją operacji krzyżowania tak aby przekazywała również informację o dopasowaniu palców.

Ponadto, w rozwiązaniach zauważone zostały również pewne artefakty (znikanie osobników dopuszczalnych, pogarszanie najlepszych rozwiązań), które mogą wskazywać na błąd



Rysunek 4: Scenariusz testowy 3

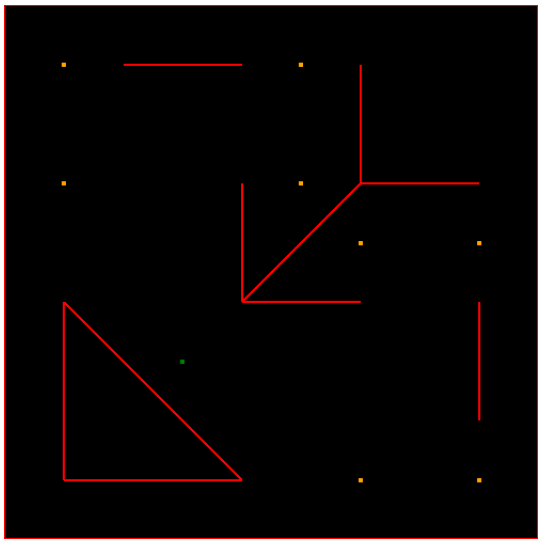


Rysunek 5: Scenariusz testowy 4

w implementacji.

Warto zwrócić także uwagę na aspekty obliczeniowe związane z problemami geometrycznymi. Zarówno wyznaczenie początkowych heurystyk, jak i obliczanie funkcji dystansu oraz błędu wymagają sprawdzania dużej liczby warunków, co przenosi ciężar obliczeń z ewolucji właściwej na aspekty dodatkowe i pozwala na wyewoluowanie w sensownym czasie mniejszej liczby generacji na mniejszych populacjach osobników.

W szczególności inne metody liczenia odległości (chodzenie po figurze lub jej otoczce wypukłej) powodowały bardzo duże spowolnienie nawet dla niewiekich domyślnych parametrów programu. Bez rozwiązania kwestii optymalizacji tych metod korzystanie z nich na większych planszach lub dla większego rozmiaru populacji byłoby niepraktyczne.



Rysunek 6: Scenariusz testowy 5

Literatura

[1] Patryk Filipiak, *Optymalizacja ewolucyjna z predykcją w dynamicznym problemie kinematyki odwrotnej*. Wrocław, Seminarium ZMN IIUWr 2011.