Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **Informatyka Stosowana (IST)**
Specjalność: **Zastosowania Specjalistycznych Technologii Informatycznych**

PRACA DYPLOMOWA MAGISTERSKA

Opracowanie algorytmu generacji grafu DSP do rozwiązania problemu syntezy dźwięku

Automated generation of signal processing graphs for sound synthesis

Mateusz Bączek

Opiekun pracy
dr inż. Maciej Hojda

Słowa kluczowe: synteza, dźwięk, graf, optymalizacja

WROCŁAW, 2022

Spis treści

Spis rysunków

Spis tabel

Spis listingów

Rozdział 1

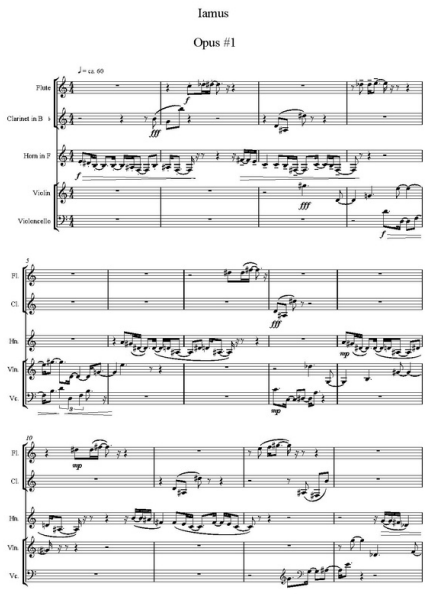
Wstęp

1.1. Wprowadzenie

Rozpowszechnione algorytmy sztucznej inteligencji wspomagające pracę inżynierów dźwięku można podzielić na dwie główne dziedziny [**?**] :

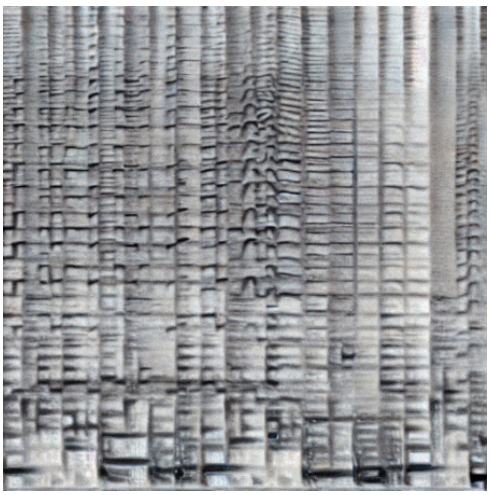
1. algorytmy generujące symboliczny zapis muzyki (nuty lub dane MIDI) (**??**),
2. algorytmy generujące gotowy plik audio (**??**).

Pierwsza grupa algorytmów znana jest już od lat 80, gdyż zagadnienie generowania zapisu symbolicznego jest mniej wymagające niż wytworzenie pełnego pliku audio. Powszechnie wykorzystywana jest w nich teoria muzyki, pozwalająca określić matematyczne relacje w rytmach, melodiach i progresjach akordów. Wiedza dotyczącą teorii muzyki pozwala na wyznaczenie możliwej przestrzeni stanów, w której generowana jest kompozycja, natomiast modele matematyczne takie jak łańcuchy *Markowa* służą za mechanizmy decyzyjne, które „nawigują” w przestrzeni stanów.



Rys. 1.1: Zapis nutowy utworu *Opus One*, wygenerowany przez komputer *Lamus*.

Druga grupa algorytmów, generująca pliki audio, rozwija się na bazie nowych możliwości, które zapewniają algorytmy wywodzące się ze *Stable Diffusion* [**?**]. Najnowsze modele generujące pliki audio zgodne z opisem tekstowym (przykładowo „`smutny jazz`" bądź „`muzyka taneczna w stylu Depeche Mode`") szkolone są w taki sam sposób jak algorytmy stable diffusion. Jednakże zamiast na obrazach artystów, modele takie jak *Stable Riffusion* [**?**] uczą się na spektrogramach, które następnie są w stanie wygenerować (**??**). Po wygenerowaniu spektrogramu przez model, jest on konwertowany do pliku audio za pomocą odwrotnej transformaty Fouriera.



Rys. 1.2: Przykładowy spektrogram wygenerowany przez algorytm *Stable Riffusion* dla danych wejściowych `funk bassline with a jazzy saxophone solo`.

Obie metody opisane w rozdziale **??** można porównać pod względem ich przydatności dla użytkownika końcowego, czyli osoby zajmującej się produkcją nagrań muzycznych. Metoda pierwsza, generowanie zapisu symbolicznego, może wydawać się mniej zaawansowana niż generowanie całych plików dźwiękowych. Jednakże, z perspektywy użytkownika, zapis symboliczny jest bardziej praktyczny, ponieważ możliwe jest zaimportowanie go do programu DAW i późniejsza modyfikacja zapisu nutowego. Obecnie dostępne modele generujące pełne nagrania z muzyką nie umożliwiają szczegółowego edytowania parametrów wygenerowanego dźwięku, ponieważ operują bardzo wysokopoziomowo – syntezują muzykę na podstawie opisu słownego.

Niniejsza praca realizuje zadanie, którego nie da się zaklasyfikować do żadnej z dwóch wyżej wymienionych (**??**) dziedzin. Wynik pracy algorytmu implementowanego w ramach pracy magisterskiej jest **gotowym elektronicznym instrumentem muzycznym**, który może być wykorzystany w programie do komponowania muzyki. Tego typu proces generowania grafów przetwarzania sygnałów dźwiękowych może być porównany z procesem projektowania instrumentu muzycznego.

Proces dynamicznego modyfikowania grafu przetwarzania sygnału jest często wykorzystywany w muzyce elektronicznej, do tworzenia dźwięków o interesującej barwie bądź dynamice. Syntezatory dźwięku dostępne na rynku często wyposażone są w *patch bay*, pozwalający na modyfikowanie grafu przepływu sygnałów wewnątrz syntezatora, bądź połączenie go z zewnętrznym sprzętem muzycznym bądź elektronicznym (**??**).

Rys. 1.3: Syntezator *Mother 32* firmy *Moog*, po prawej stronie widoczny jest *patch bay* z podłączonymi przewodami, które zmieniają konfigurację połączeń między układami generującymi i przetwarzającymi sygnał dźwiękowy.

1.2. Cel pracy

Celem pracy jest zbadanie, czy algorytmy optymalizacyjne są w stanie wytworzyć graf przetwarzania sygnałów audio, który wykona syntezę próbki dźwięku zadanej przez użytkownika. Problem poruszany w pracy można zakwalifikować do grupy zagadnień związanych z pojęciem *computer-aided design* w dziedzienie inżynierii dźwięku. Docelowo zaimplementowany algorytm będzie automatyzował pracę inżyniera dźwięku, tworząc i konfigurując grafy przetwarzania sygnałów dźwiękowych, dostępne w programach typu *digital audio workstation* **??**. Badania obejmują dwa zagadnienia:

1. metody generowania grafu przetwarzania sygnałów oraz późniejszej modyfikacji grafu,
2. dobór funkcji celu, na podstawie której algorytm optymalizujący będzie przeszukiwał możliwą przestrzeń grafów przetwarzania sygnałów.

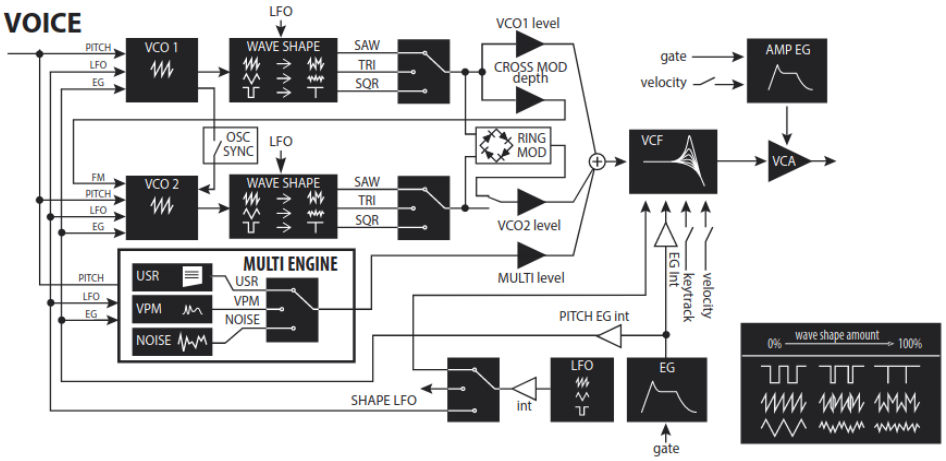
Pierwsze zagadnienie sprowadza się do przetestowania szeregu algorytmów pozwalających na wygenerowanie grafu przetwarzania sygnałów DSP oraz ich modyfikację. Przykładem modyfikacji grafu może być wprowadzanie w nim losowych zmian lub krzyżowanie dwóch grafów DSP w przypadku wykorzystania algorytmu genetycznego. Graf przetwarzania sygnałów można opisać jako zbiór połączonych węzłów generujących i przetwarzających sygnał dźwiękowy. Każdy węzeł można opisać poprzez:

1. zbiór wejść,
2. zbiór wyjść,
3. operację matematyczną, wykonywaną na sygnale.

Pełny graf przetwarzania można opisać za pomocą zbioru węzłów oraz macierzy połączeń między węzłami:
N - liczba węzłów,
$i_j = [p_1, p_2, \ldots, p_n]$ – Zbiór wejść (*inputs*) j-go węzła,
$o_j = [p_1, p_2, \ldots, p_n]$ – Zbiór wyjść (*outputs*) j-go węzła,
$f_i(x)$ – operacja wykonywana na sygnale przez i-ty węzeł.

$C = [\{o_{(j,k)}, i_{(l,m)}\}, \ldots]$ – zbiór połączeń między węzłami, opisujący, które k-te wyjście j-go węzła podłączone jest do którego m-go wejścia l-go węzła.

Nie wszystkie wejścia w grafie muszą wyć podłączone do któregoś z wyjść. Wejście, które nie zostało nigdzie podłączone przyjmuje jako wartość parametr liczbowy optymalizowany później w funkcji celu **??**. W przypadku schematu **??** takimi „wolnymi" wejściami są przykładowo sygnał określający wysokość dźwięku oraz parametry określające parametry generatora obwiedni (*EG*).



Rys. 1.4: Diagram blokowy pojedynczego głosu w syntezatorze *Minilogue xd* firmy *Korg* [**?**].

Dla powszechnie wykorzystywanego w analogowych syntezatorach subtraktywnych schematu przetwarzania sygnałów (**??**) można wyróżnić przykładowe węzły:

Oscylator (*VCO*):

1. Wejścia:
   - częstotliwość,
   - kształt fali.
2. Wyjścia:
   - wygenerowany sygnał.

Filtr (*VCF*):

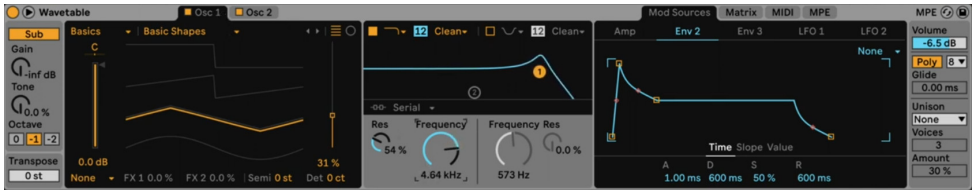
1. Wejścia:
   - częstotliwość odcięcia,
   - rezonans.
2. Wyjścia:
   - przefiltrowany sygnał.

Drugie zagadnienie obejmuje przetestowanie szeregu algorytmów, które można wykorzystać jako funkcję celu, która będzie optymalizowana poprzez „dostrajanie" grafu przetwarzania sygnałów dźwiękowych.

Funkcja celu, oceniająca, jak sygnał wygenerowany ($\bar{x}$) przez algorytm jest bliski sygnałowi docelowemu (x) może zostać przedstawiona w następujący sposób:

$$F(x, \bar{x}) = q \tag{1.1}$$

Następnie, dla danego układu N węzłów przetwarzania oraz dla macierzy połączeń C, należy rozwiązać następujący problem optymalizacji, należy rozwiązać problem maksymalizacji funkcji opisanej równaniem **??** dla parametrów wszystkich wejść i_j oraz o_j, które nie są połączone bezpośrednio pomiędzy węzłami.



Rys. 1.5: Zbiór parametrów konfigurujących przykładowy syntezator dźwięku w programie *Ableton*

1.3. Zakres pracy, plan badań

1.3.1. Metody generowania grafu przetwarzania sygnałów oraz późniejsza modyfikacja grafu

Głównym problemem przy generowaniu grafu przetwarzania sygnałów są ograniczenia nałożone na strukturę grafu, które należy spełnić, by graf był logicznie interpretowalny jako łańcuch przetwarzania sygnałów. Graf musi być grafem skierowanym, który nie zawiera pętli o dodatnim sprzężeniu zwrotnym (lub nie zawiera ich wcale, co można założyć dla uproszczenia problemu). Struktura grafu powinna być możliwie jak najbardziej przejrzysta dla użytkownika. Automatyczna ewolucja może dążyć w kierunku wykorzystania nadmiarowej liczby bloków przetwarzania sygnału, jeśli funkcja celu nie będzie zawierała kary za zbyt złożone grafy. Podobne prace [**?**] wykorzystują podejście oparte o *mixed-typed carthesian genetic programming*, które będzie punktem startowym dla pracy. Finalnie, badania dążą do wyznaczenia algorytmu o następujący właściwościach:

1. algorytm generuje grafy będące logicznie spójnymi łańcuchami przetwarzania dźwięku (skierowany, bez pętli o dodatnim sprzężeniu zwrotnym w natężeniu sygnału),
2. algorytm maksymalizuje wykorzystanie poszczególnych bloków przetwarzania w grafie, co minimalizuje finalny rozmiar grafu, czyniąc go bardziej czytelnym,
3. generowany graf posiada reprezentację umożliwiającą wykonanie krzyżowania dwóch grafów przetwarzania sygnału. Graf będący wynikiem krzyżowania nadal musi być poprawnym grafem przetwarzania sygnału.

Elementami grafu przetwarzania sygnałów są używane powszechnie w syntezie dźwięku algorytmy:

1. modulacja FM [**?**] [**?**],
2. synteza subtraktywna [**?**] [**?**],
3. algorytmy *physical modeling* [**?**] [**?**],
4. symulacja efektu pogłosu/echa [**?**] [**?**].

1.3.2. Dobór funkcji błędu – różnica między wygenerowanym a docelowym sygnałem dźwiękowym

Funkcja celu poszukiwana w ramach projektu musi określać, jak dobrze sygnał wygenerowany przez graf przetwarzania sygnałów pokrywa się z sygnałem docelowym. Porównanie sygnałów musi skupiać się na cechach sygnału, które są najbardziej słyszalne dla ludzkiego ucha. Jednocześnie funkcja nie powinna „karać” sygnałów, które są względem siebie przesunięte w fazie. Wśród algorytmów, które zostały wybrane do przetestowania w ramach projektów zawarte są:

1. algorytmy porównywania sygnałów oparte o transformatę Fouriera [**?**] [**?**],

2. techniki wykorzystywane do generowania „cyfrowych podpisów” sygnałów dźwiękowych (*sound fingerprinting*) [**?**],
3. algorytmy wykrywające spadek jakości dźwięku z perspektywy psychoakustycznej [**?**] [**?**].

Dodatek A

Instrukcja wdrożeniowa

Jeśli praca skończyła się wykonaniem jakiegoś oprogramowania, to w dodatku powinna pojawić się instrukcja wdrożeniowa (o tym jak skompilować/zainstalować to oprogramowanie). Przydałoby się również krótkie „*how to*” (jak uruchomić system i coś w nim zrobić – zademonstrowane na jakimś najprostszym przypadku użycia). Można z tego zrobić osobny dodatek.
`cargo run` i jazda

Dodatek B

Opis załączonej płyty CD/DVD

Tutaj jest miejsce na zamieszczenie opisu zawartości załączonej płyty. Opis ten jest redagowany przed załadowaniem pracy do systemy APD USOS, a więc w chwili, gdy nieznana jest jeszcze nazwa, jaką system ten wygeneruje dla załadowanego pliku. Dlatego też redagując treść tego dodatku dobrze jest stosować ogólniki typu: „Na płycie zamieszczono dokument `pdf` z niniejszej tekstem pracy" – bez wskazywania nazwy tego pliku.

Dawniej obowiązywała reguła, by nazywać dokumenty według wzorca `W04_[nr albumu]_[rok kalendarzowy]_[rodzaj pracy]`, gdzie `rok kalendarzowy` odnosił się do roku realizacji kursu „Praca dyplomowa", a nie roku obrony. Przykładowo wzorzec nazwy dla pracy dyplomowej inżynierskiej w konkretnym przypadku wyglądał tak: `W04_123456_2015_praca inżynierska.pdf`, Takie nazwy utrwalane były w systemie składania prac dyplomowych. Obecnie działa to już inaczej.