Lehrstuhl VIII - Musik als Daten

Playlists

Raffael Joliet

Übersicht

- Definitionen / Einführung
- Playlistgenerierung anhand von inhaltlicher Ähnlichkeit
  - kombinatorisch mit Randbedingungen (CSP)
  - mittels BILP
  - Adaptive Suche
  - durch Ordnung aller Titel (TSP)
  - Dynamisch mit Feedback
- Playlistgenerierung durch kollaboratives Filtern
  - Flycasting
- Halbautomatische und visuelle Playlistgenerierung
- Zusammenfassung

Definition: Playlist

- *„sequences of music titles“* (Pachet, 2002, „Scaling Up Music Playlist Generation“)
- *„a contrained sequence of music pieces that satisfy users preferences“* (Alghoniemy, 2001, „A Network Flow Model For Playlist Generation“)

Wunsch nach Wiederholung
Wunsch nach Abwechselung
Wunsch nach einer guten Durchdringung der zur Auswahl stehenden Musikstücke

Motivation

- MIR funktioniert im Wesentlichen

PROBLEME:

- Einzelne Titel selten gewünscht
- Sequenzen von Liedern vermitteln etwas anderes als eine Mengen von Titeln.

Einführung

- Verschiedenste Ansätze →
  - Die Aufgabe „Playlistgenerierung“ hat keine scharfe Abgrenzung. Beispiel: Playlists vs. Empfehlungen
  - Die Güte einer Playlist ist nur durch Evaluationen bestimmbar. Keine objektiven Kriterien.
- Erste Veröffentlichen: 2000 von Pachet u.a.
- Trend: Von kombinatorischen Ansätzen hin zu interaktiven, grafischen Lösungen
- Im folgenden: Eine Auswahl von Ansätzen.

Kombinatorische Playlistgenerierung

- Frühester Ansatz für das *„music selection problem“* → *„match user preferences, provide users with new music, and exploit the catalogue in an optimal fashion“* (Pachet, Roy, Cazaly, „A Combinatorial Approach to Contetn-based Music Selection“)
- *„we do not know any other attempt at generating sequences of multimedia data.“*
- Annahmen über die Wünsche des Benutzers:
  - Wunsch nach Wiederholung (bisher: MIR)
  - Wunsch nach Überraschung (bisher: CF)
  - Ausschöpfen des Kataloges

Kombinatorische Playlistgenerierung(2)

- Ansatz: Manuelle Vorgabe von Randbedingungen:
  - User preferences: (At least 30% female-type voice)
  - Constrains on the coherence of the sequence
  - Constrains on the exploitation of the catalogue
- Basiert auf einer Datenbank von Musikstücken, die (manuell) mit Metadaten versehen sind
- Die Werte der Attribute sind (halb-) geordnet bzw. Ähnlichkeiten sind definiert

Kombinatorische Playlistgenerierung(3)

- Technik: Constraint Satisfaction Programming (CSP)
  - Playlist als Menge von Items (Variable mit dem Katalog als Wertebereich)
  - Jede Variable v_i hat j zugehörige Attribute v_i^j.
  - Formulierung der Randbedingungen:
    - S(a, b, j, similar(,)) = For every v_i, $i \in [a, b-1]$, similar(v_i^j, v_{i+1}^j)
    - D(I, j) = All items v_i, $i \in I$, have pairwise different values for j.
  - Similarity constraints, difference constraints, cardinality constraints (on items, on attribute values)

Playlistgenerierung durch BILP

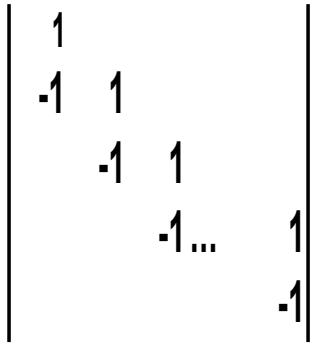
- Nach Alghoniemy: vorheriges Problem **NP-hard**, da der gesamte Lösungsraum (alle Kombinationen) erforscht werden muss.
- Zur Verringerung der Komplexität:
  - „approximation“
  - Umformulierung in lineare Gleichungen (Vektoren/Matrizen)
  - Aufstellen eines BILP

Playlistgenerierung durch BILP (2)

- Voraussetzungen: Musik ist schon kategorisiert und mit Metadaten versehen
- M Tracks als Spaltenvektoren mit N (binären) Attributen sei binäre Matrix A
- Gesucht: Matrix R (N x L) mit L Ergebnistracks
- Zwei Arten von Constraints:
  - „coherence constraints“
  - „absolute constraints“

Constraints

- Coherence constraints
  - Sichern die Kontinuität zwischen den Tracks
  - werden als Differenzmatrix D modelliert: (L x (L-1))



- Absolute constraints
  - Legen die minimale und maximale Anzahl des Vorkommens jedes Attributes fest
  - als 2 Spaltenvektoren C_{min} und C_{max}

Optimization

- Optimiere: $E = \arg\min_E \| AED \|_1$

Maximale Zeilensumme von AED

Suche Permutation E

Zur Auswahl stehende Stücke

Kontinuitäts-Contraints

- Nebenbedingungen: $AE\,\mathbf{1} \geq C_{min}$, $AE\,\mathbf{1} \leq C_{max}$
- Zur Berechnung, aufteilen in zwei Teile:
  - Finden einer Menge von Titeln, die den absoluten Contraints genügen
  - Finden einer Reihenfolge der Titel in der Playlist

BILP

- Sei **x** ein Vektor der Länge M, der in Zeile k eine 1 hat, falls Track k zur Playlist gehört, sonst 0.
- Sei $\mathbf{f} = [\, ||\, a_1\, ||_1 \;\; ||\, a_2\, ||_1 \;\; \ldots \;\; ||\, a_M\, ||_1 g\,]^T$ mit a_i als der i-te Spalte der Datenbankmatrix A
- Dann entspricht unser Problem (ohne Reihenfolge):

Minimiere: $\min_x \mathbf{f}^T\mathbf{x}$

unter: $A\mathbf{x} \geq C_{min}$ $\mathbf{1}^T\mathbf{x} = \boldsymbol{L}$

$A\mathbf{x} \leq C_{max}$ $\mathbf{x}_k \in \{0,1\}$

Reihenfolge der Playlist

- Finde unter den *L!* Möglichen Playlist jene mit höchster Kontinuität → Schwer
- Benutze die implizite Nummerierung der Datenbankmatrix A
- Führe das LP mehrfach aus mit zyklisch geshifteten Matrixspalten (vereinfacht durch branch and bound)
- Berechne für jeden Durchgang die gewünschte Ähnlichkeit
- Ergebnis: lokales Maximum der Ähnlichkeiten

Zusammenfassung

- Möglichkeit zur Komplexitätsreduktion auf Kosten der Genauigkeit
- Weitere Arbeit der Autoren:
  - Zusätzliche Vorgabe von Start- und Zieltrack der Playlist
  - Modellierung über Netzwerkfluss, Lösung mit ILP
- **Performance:** 400 Titel mit 10 binären Attributen. Generierung eine 7 Titel langen Playlist: 35 Sekunden auf Ultra 5 Plattform (80 zyklische Shifts)

Adaptive Suche

- Antwort von Pachet und Aucouturier:
  - Kritik an der Eingeschränktheit mit binären Attributen
  - Kritik an der Performance
  - Erkenntnis: Das Problem ist NP-hard und bedarf anderer Ansätze.
- 2002, Präsentation eines Ansatzes, der alle bis dato vorgestellten Ansätze subsumieren soll.
- Idee: Flexible Constraints, die als Kostenfunktionen modelliert werden.

Adaptive Suche (2)

- Playlist: Sequenz von n Variablen, v_1, ... , v_n mit Werten x_i
- Kostenfunktionen:
  - *Cost (v_i, C)* gibt an, wie schlecht Variable v_i Contraint C erfüllt
  - *Cost (v_i)* gewichtetes Mittel über die Cost (v_i, C)
- Beispiel: alle Titel unterschiedlich:

```
AllDifferentCt.cost()

Return 1 – number of different values in the sequence
  divided by the size of the sequence
```

Adaptive Suche (3)

- Weiteres Beispiel: „the playlist should contain at least 60% of instrumental titles“:

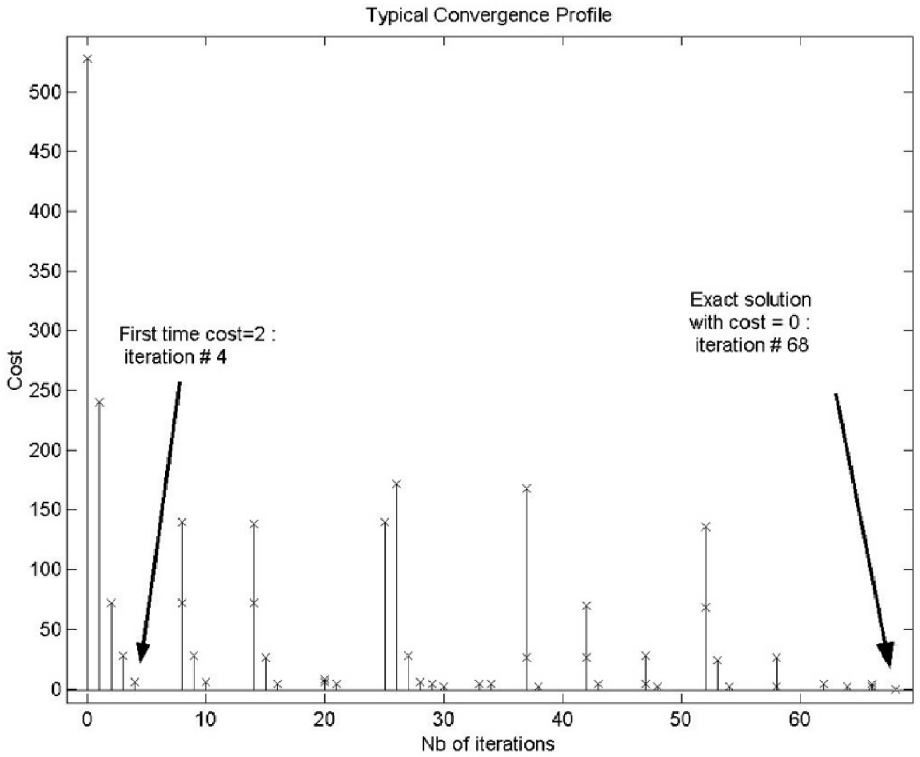
```
CardinalityCt.cost()

Return abs(actual number of values that have the wished
   attribute – wisched number of values) divided by the
   size of the sequence
```

- Vorfilterung zur Reduktion des Suchraumes, falls für bestimmte Playlistpositionen nicht alle Titel in Frage kommen. (Datenbankmethoden)

Adaptive Suche (4)

- Eigentlicher Such-Algortihmus:
  - Starte mit einer zufälligen Playlist
  - Berechne die Kosten für die gesammte Liste durch Aufsummieren der *Cost* (v_i)
  - Wiederhole, bis die Kosten unter einen Schwellwert fallen:
    - Berechne die Kostenfunktionen für jede Variable
    - Finde die Variable mit den größten Kosten
    - Finde für diese aus dem Suchraum die Belegung mit den niedrigsten Kosten für die gesamte Playlist
    - Weise diese Belegung der Variable zu
- Zusätzlich: Behandlung von lokalen Minima (simulated annealing)

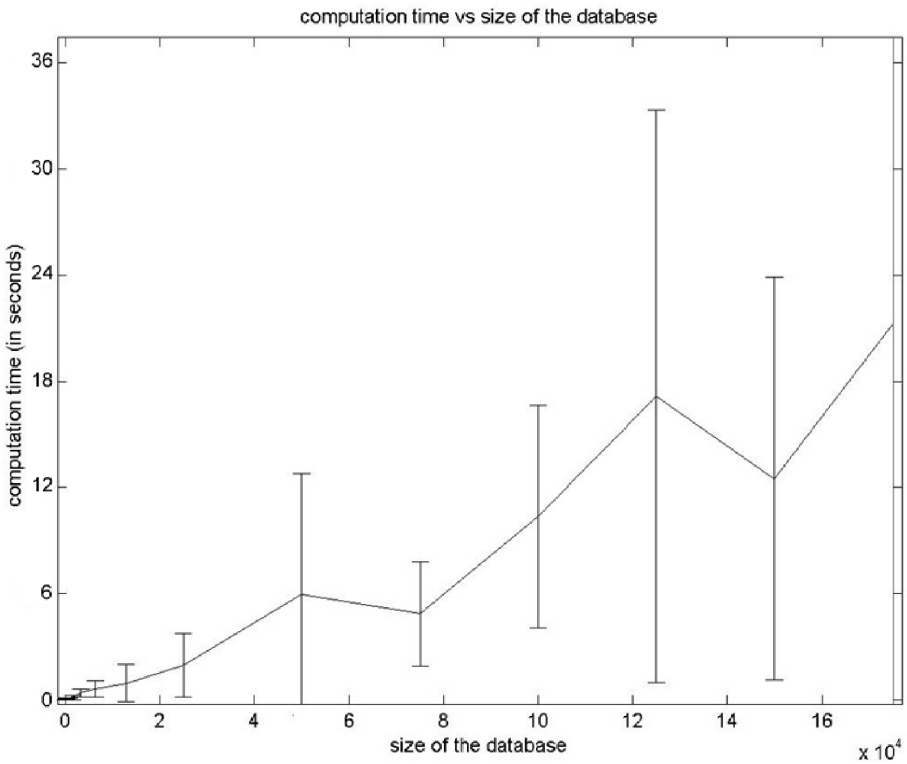
Adaptive Suche (5)



Convergence of the Costs
„Scaling up Music playlist Generation“

- Typische Profil der Kosten bezogen auf die Iterationen
- Vorteil: zu jedem Zeitpunkt Resultate
- Ziemlich schnell gute, aber nicht exakte Resultate

Adaptive Suche (6)



Convergence time vs. The the DB-size
„Scaling up Music playlist Generation“

- Geschwindigkeit um viele Größenordnungen besser
- Exaktes Resultat einer Playlist mit 10 Titeln aus 17.000 Titeln und mehr als 6 Constraints: 4 Sekunden
- Kaum Zunahme bei längeren Listen
- Linear mit der Datenbankgröße

Ansatz für mobile Player

- Finde eine Ordnung auf den Liedern auf z.B. einem MP3-Stick.
- Spiele einen Ausschnitt aus diesem Pfad
- Kriterium: inhaltliche Ähnlichkeit gemessen an den MFCC-Koeffizienten
- Benutze Heuristiken von TSP (Greedy, Spannbaum, Lin-Kerningham, SOM)

Dynamische Playlists

- Überschneidung mit Musikempfehlungssystemen
- Die Playlist wird anhand von z.B. User-Feedback dynamisch angepasst
- Konzept nach Pampalk u. A.:
  - Gegeben: Startsong
  - System mit einem Skip-Button
  - Benutzer überspringt einen Titel, wenn er ihn nicht mag.
  - Ähnlichkeit anhand von „spectral similarity“ und „fluctuation patterns“

Dynamische Playlists (2)

- Mögliche Songs sind alle, außer bereits gespielte und übersprungene
- 4 Heuristiken zur Playlistgenerierung:

  A) Die nächsten Nachbarn bezüglich der gegebenen Ähnlichkeit werden zu einer Playlist zusammengefasst. *Statisch*

  B) Es wird *dynamisch* immer der Song als nächstes gewählt, welcher am nächsten zum letzten akzeptierten liegt.

  C) Der Song, der zu irgendeinem akzeptierten Song am nächsten liegt wird gespielt

  D) Verwalte eine Menge S. Füge Song s hinzu, falls seine Nähe zu einem akzeptierten Song größer ist als die zu einem abgelehnten. Spiele den Song aus S mit dem kleinsten Abstand zu einem akzeptierten Song.

Dynamische Playlists (3)

- Teste diese Heuristiken unter folgenden 3 Anwendungsfälle:
  1. Der Benutzer möchte nahe beim Startsong bleiben. Songs außerhalb dessen Genre werden übersprungen.
  2. Wie 1 aber mit zufälligem Startsong (aus dem gewünschten Genre)
  3. Der Benutzer bevorzugt einen Verlauf von Genre A zu B.

Ergebnisse

- Enorme Reduzierung der Skips (im Vergleich zu einer zufälligen Playlist (UC-A: 1/10)
- Anwendungsfall 3 funktioniert teilweise besonders gut, liefert aber zum Teil auch miserable Ergebnisse
- Großer Einfluss des Ähnlichkeitskriteriums!

Was gibt es noch?

- Viele Ideen, in denen an die Möglichkeit zur Playlistgenerierung gedacht wurde.

Übersicht

- Definitionen / Einführung
- Playlistgenerierung anhand von inhaltlicher Ähnlichkeit
  - kombinatorisch mit Randbedingungen (CSP)
  - mittels BILP
  - Adaptive Suche
  - durch Ordnung aller Titel (TSP)
  - Dynamisch mit Feedback
- Playlistgenerierung durch kollaboratives Filtern
  - Flycasting
- Halbautomatische und visuelle Playlistgenerierung
- Zusammenfassung

Flycasting

- „CF is based on the idea that there are patterns in tasts: **tastes are not distributed uniformly**“ (Pachet, Roy, Cazaly, „A Combinatorial Approach to Contetn-based Music Selection“)
- Ziel: Generiere die Playlist für ein Online-Radio abhängig vom Kreis der Zuhörer
- Idee: Bewerte die Beliebtheit von vom Publikum nicht gefragten Künstlern danach, ob es gemeinsame Auftreten zwischen gefragten und zu bewertenden Künstlern gibt.

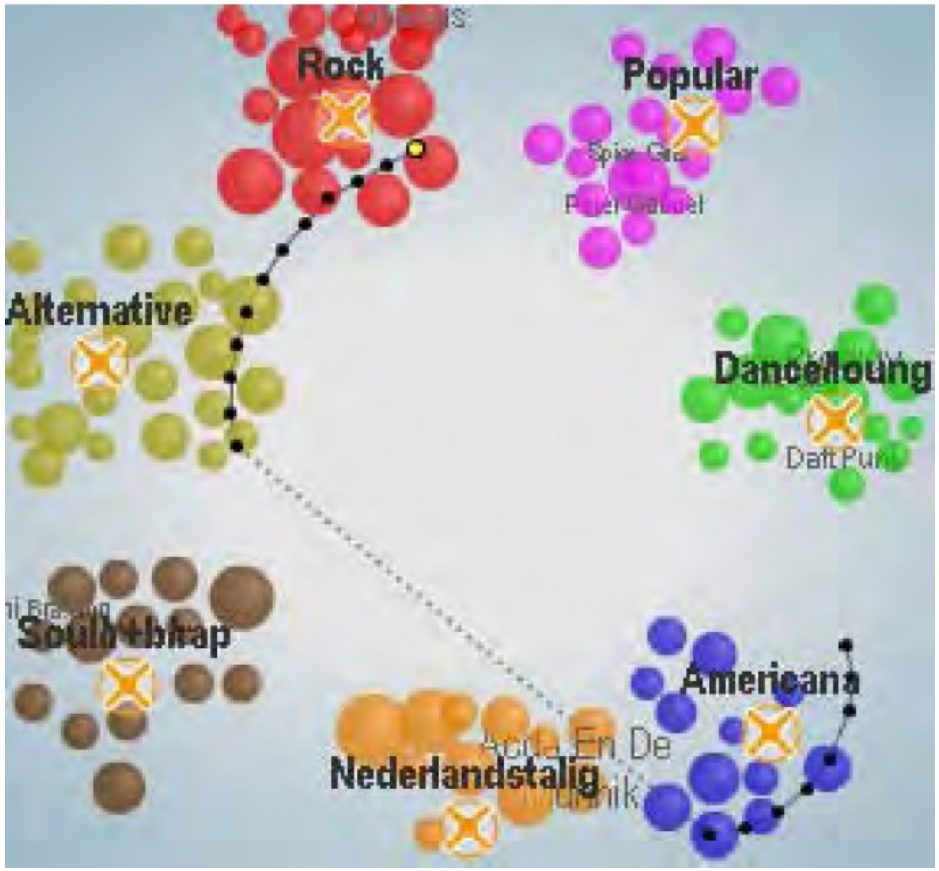
Interaktive Playlistgenerierung

- Systeme zur Unterstützung einer manuellen Playlisterstellung
- Hilfe durch kontextsensitive, inhaltsbezogene Vorschläge
- Meist für Experten entwickelt.

SatisFly.
„User Avaluation of a new interactive Playlist generation concept“, Pauwes, Wijdeven, 2005

Visuelle Playlistgenerierung

- Kontrolle auf einer höheren Ebene
- Artist Map (SOM)
- Datenunabhängige Playlists
- Interaktive Manipulierbarkeit



Playlist path
„Visual Playlist Generation on the Artist Map“, Gulik, Vignoli, 2005

Zusammenfassung

- Verschiedenste Methoden und Motivationen
- Schwierige Probleme (NP-hard)
- Unterschiedlichste Auffassungen einer guten Playlist
- Unterschiedlichste Anwendungen
- Konzepte rufen bei Benutzern sehr ambivalente Reaktionen hervor