University of Southern Denmark

Department of Mathematics and Computer Science

Master Thesis – 31/12-2017

# Computational Synthesis Planning Using Big Data

Computational Syntese planlægning ved hjælp af big data

*Author:*
Henrik Schulz

*Supervisors:*
Daniel Merkle

## Abstract

## Resumé

## Contents

## 1 Acknowledgements

## 2 Introduction

kkk [2]

### 2.1 Overview

Hvad indeholder hver sektion??

## 3 Preliminaries

This section contains definitions that will be used throughout this paper. It is assumed that the reader have a basic understanding of graph theory.

**Hypergraphs**
A directed hypergraph $h$ is a set of $V$ of vertices and a set $E$ of hyperedges, where each hyperedge $e = (T(e), H(e))$ is an ordered pair of non-empty multisets of vertices. The set $T(e)$ is denoted as the tail of the hyperedge and $H(e)$ is the head. If $|H(e)| = 1$ then the hyperedge is denoted as a B-hyperedge. If all edges in the hypergraph is B-hyperedges, then the graph is denoted a B-hypergraph. This paper will only consider hypergraphs that are B-hypergraphs. A hypergraph $H' = (V', E')$ is a subhypergraph of $H = (V, E)$ if $V' \subset V$ and $E' \subset E$.[3]

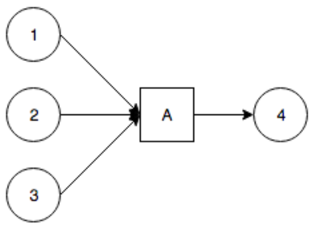

Fig. 1: Example of hyperedge $A$. $T(A) = \{1, 2, 3\}$, $H(A) = \{4\}$

**Hyperpaths**
A path $P_{st}$ from $s$ to $t$ in a B-hypergraph is a sequence $P_{st} = \langle e_1, e_2, e_3, ..., e_q \rangle$ of B-hyperedges such that $s \in T(e_1)$ and $t = H(e_q)$ and $H(e_i) \in T(e_{i+1})$ for $i = 1..q-1$. Its length $|P_{st}|$ is the number $q$ of hyperedges. If $t \in T(e_1)$, then $P_{st}$ is a cycle. A hypergraph is acyclic if it does not contain any cycles. [3]

## 4 Finding The K-Best Synthetic Plans

kkk [5]
kkkk [3]

## 4.1 Yen's Algorithm

Yen's algorithm is an algorithm that computes the K-shortest paths for a graph with non-negative edges. It was publish in 1971 and uses any shortest path algorithm to find the best path and then proceeds to find the $K-1$ deviation of the best path.
It starts out by finding the best path using a shortest path algorithm. Once the best path have been found it uses the path to find all the potential next best paths by fixing and removing edges in the graph.
By using the same first vertex as the original path but removing the first edge, it forces the shortest path algorithm to take another route through the graph and thereby creating a potential second best path. This is added to the list of potential paths and the algorithm can continue to derive other paths from the best path. By fixing the first edge in the previous best plan, Yen's algorithm forces the shortest path algorithm to take the first edge which it now shares with the best path. However, now the algorithm have removed the second edge from the original path and once again forces the shortest path algorithm to find alternative routes. This process is then repeated until we reach the next to last vertex in the best path.
By sorting the list of potential paths, it has the second best path at the start of the list and it can add it to the final list of best path. The algorithm then repeats on the second best path to find the third best path. This is done until all K-best path have been found.

We use the principles from Yen's algorithm to make our own algorithm that will work on hypergraphs. To handle the problem of generating all derived paths from our best path in our hypergraph, we use a method called Backwards-Branching.[4]
However, this algorithm have a problem when working on a larger hypergraph. It demands that each time we make alterations on the hypergraph we have to copy all vertices and all arcs except

Algorithm 1: Backwards Branching for B-Hypergraph using overlay

```
function BackwardsBranching(Path, Overlay)
    List B = $\emptyset$
    // q = Path length
    for i = 1 to q do
        //remove i'th hyperedge from Path in overlay
        Set Overlay[Path[i]] to false
        //fix the backtree
        for j = i downto 1 do
            Get compound C <− Path[j].head
            for each reaction into C
                Set Overlay[reaction.id] to false
            Set Overlay[Path[j]] to true
        endfor
        B = B $\cup$ $\{Overlay\}$
    endfor
    return B
endfunction
```

Algorithm 2: K-Shortest Paths Algorithm in B-Hypergraph

```
function YenHyp(s, t, K)
    L = new heap with elements (overlay, maxYield)
    A = List of shortest paths
    //(Graph is default overlay (all true))
    $\pi$ = shortestPath(Graph, s,t)
    Insert (Graph, $\pi$) into L
    for k = 1 to K do
        if L = $\emptyset$
            Break
        endif
        (Overlay$'$, $\pi'$) = L.pop
        for all Overlay$^i$ in BackwardBranching((Overlay$'$,$\pi'$)) do
            $\pi^i$ = shortestPath(Overlay$^i$, s, t)
            if $\pi^i$ is complete
                Insert( $H^i$, $\pi^i$) into L
            endif
        endfor
    endfor
    return A
endfunction
```

## 4.2 Back-Branch

#### 4.2.1 Overlay

d[6]

## 5 Shortest Path

## 5.1 Dynamic Approach

kkk [5]
kkkk [3]

#### 5.1.1 Approach

#### 5.1.2 Testing

#### 5.1.3 Problems

## 5.2 Nielsens Algorithm

k [4]

#### 5.2.1 Approach

#### 5.2.2 Testing

#### 5.2.3 Optimizing

# 6 Work With Beilstein Data

## 6.1 The Graph

## 6.2 Testing

### 6.2.1 Strychnine

### 6.2.2 Compound 2

### 6.2.3 Compound 3

kkkk [1]

# 7 Konklusion

## Books

[1] R. W. Hoffmann, *Elements of Synthesis Planning.* Springer Berlin Heidelberg, 2009.

## Articles

[2] S. Szymkuc, E. P. Gajewska, T. Klucznik, K. Molga, P. Dittwald, M. Startek, M. Bajczyk, and B. A. Grzybowski, "Computer-assisted synthetic planning: The end of the beginning", *Angewandte Chemie International Edition*, no. 55, pp. 5904–5937, 2016.

[3] R. Fagerberg, C. Flamm, R. Kianian, D. Merkle, and P. F. Stadler, "Finding the k best synthesis plans", 2017, Unpublished Article.

[4] L. R. Nielsen, K. A. Andersen, and D. Pretolani, "Finding the k shortest hyperpaths: Algorithms and applications", 2002.

## Other

[5] C. G. Lützen and D. F. Johansen, "A computational and mathematical approach to synthesis planning", Master's thesis, University of Southern Denmark, 2015.

[6] Sep. 2017. [Online]. Available: `http://en.cppreference.com/w/cpp/container/vector_bool`.

## 8 Appendiks