OptLearn algorithm UL

Technical report

Andrej Košir, Gregor Strle, Urban Burnik, Janez Zaletelj

1 Introduction

This technical report covers partner University of Ljubljana's contributed OptLearn algorithm. The aim is to explain 1. Background and basic approach, 2. Assumptions, 3. Algorithm design and outline with decisions made, and 4. Conclusion remarks.

2 Background and basic approach

The goal of OptLearn algorithm is to recommend a sequence of questions to a student to maximize her selected learning indicators in a given amount of time (effort). We decided to base our algorithm on automatic concept maps generation since it allow us 1. Use learning indicator as a cost function explicitly, and 2. Allow for direct content-wise impact on recommendations. Therefore, We split the algorithm into

1. Concept map generation and manipulation;
2. Question recommendation based on the concept map.

There are several approaches to concept map generation. We started by incorporating the approach of Chen [1], [2] where the assumption is that all wrong-answered questions associate with one group of concepts and all right-answered questions associate with the other group of questions. We implemented the approach but on our test data, it did not provide stable results. Motivation from [3] also did not provide stable results. The alternative from [4] was promising but not applicable to the concept map-based approach. Next, we developed an approach based on keywords. This approach allows us to use existing keywords or to use Natural language processing to extract new ones. This approach yields good results. We also enabled teacher-generated concept map generation.

Since concept maps are contributed from different sources, we implemented a merging and simplification algorithm.

3 Assumptions and decisions

This is a list of conceptual issues that need to be decided in the algorithm implementation:

Question, issue	Decision, comment
Should all questions be recommended?	No. A good student can master the topic by answering just a subset of questions.
Can the questions be repeated?	Yes, especially those answered wrongly in prior sessions. No repeated questions inside the session.
Can a concept be mastered by not answering all its questions?	Yes, when enough questions are answered, it is good for a student to move on to the next concept.
Should all concepts be mastered?	Yes, by answering enough questions correctly.
Should students go back to answer questions answered wrongly, that is going back to previous concept?	Yes, such questions are the proof he did not master the topic.
Should students revisit the same concept?	Yes, when necessary to answer enough questions to master the topic.
What are the stopping criteria indicating the student mastered the topic?	See the stopping criteria below.

Note that we found out that repeating questions the student has answered wrongly might be a useful strategy.

Algorithm parametrization : we narrowed down the algorithm parametrization to four parameters only:

Algorithm 1 Parametrisation

```
qrec_conf =
'conc_compl_T':1.0,          ▷ Minimal ratio of completed concepts
'conc_succ_T':0.8,           ▷ Minimal ratio of success
'tot_compl_T':0.8,           ▷ Minimal ratio of completed questions in the concept
'no_q_code': -1              ▷ Code for "no more questions available"
```

Stopping criteria for a given student are regulated by the algorithm configuration:

1. Mastering concept: have student mastered the concept
   - 1.1 She completed at least conc_compl_T questions
   - 1.2 She achieved correctness of a given concept at least conc_succ_T
2. Mastering topic: at least conc_comp_T concepts mastered

To achieve this, we need to:

1. Repeatedly visit concepts to achieve 1.2 and 2.
2. Repeat questions that are answered wrongly

4 Algorithm design

In this section, brief outlines of the underlying algorithms are given, and referred to algorithm implementations are given in the next section.

4.1 Concept maps

A concept map is defined as a directed graph of concepts. In this frame, concepts are groups of questions that should be associated together during the student's learning process. Directed links indicate a time ordering by which students should study the content (in this particular case, answering questions in self-assessment sessions). An example of a concept map generated by the proposed algorithm is given in Fig. 1.

Our approach to automatic concept map generation is based on the following components described in the sequel.

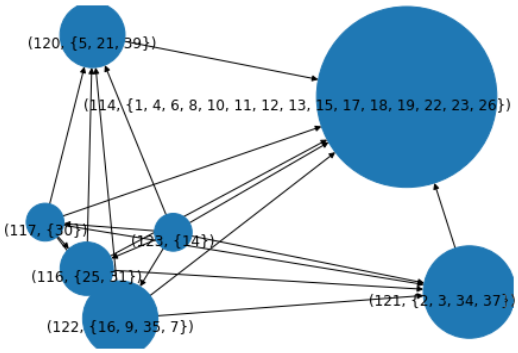

Figure 1: An example of a concept map where concepts are sets of questions and directed links indicate the order of answering them.

4.1.1 Concepts maps from various perspectives

We decided to address the concept map generation problem from several perspectives which are its dependence on (1) Available course data, (2) Teacher's (human expert) perspective, (3) Student group, and (4) This particular student. This led us to design Concept map generation algorithms by the following components:

1. Initial concept map: simple linear ordering of concepts
2. Content keywords-based concept map: keywords are the core of concepts (Alg 2).
3. Student history-based concept map: following the idea from Chen et al [1].
4. Concept map generated by the expert (teacher): The teacher can enter the structure of the content.

4.1.2 Concept maps weighted merging (Alg 3)

A primary concept map is generated (for instance, from keywords) and additional concept maps are merged according to preselected weights. A concept map is improved by merging it with another concept map according to preselected weights.

4.1.3 Concept map simplification (Alg 4)

Concept map generation and merging typically lead to complex weighted graphs which need to be simplified in order to provide better results.

4.2 Recommender system as a Concept map walk (Alg 5)

The goal of the recommendation algorithm is to calculate a personalized learning path for a student, based on content structure, her learning history, and success. A learning path is an ordered sequence of tasks (problems) that a student is expected to successfully solve to master a topic of the course.

4.2.1 Recommender system algorithm

The recommendation algorithm simulates the natural learning path of a student, progressing from basic concepts and easier tasks toward more elaborate concepts and more difficult tasks. The hierarchical relation between concepts and related questions is given in a concept map. Further, we assume that there is a large pool of questions, possibly associated with multiple concepts. So each concept might have a large pool of associated questions, where some might be very similar. In order to master a concept, a student needs to correctly solve a subset of the questions, and then he/she is allowed to progress to the next concept.

In our basic scheme, we assume a fixed threshold of correctly answered questions of the concept, which indicates successful completion of the topic. However, more elaborate concept-dependent thresholds could be utilized.

4.2.2 Learner success score

For each concept of the course, a success score is calculated from previous answers, according to the criteria:

- completeness: % of correctly answered questions of the concept;
- correctness: % of correct answers (vs. all answered questions).

Then, all questions of the concept are sorted into three categories: unanswered questions, correctly and incorrectly answered.

Generation of the next question relies on parameters minimum correctness, and minimum completeness (to move to the next concept) following the rules:

- If the concept is not yet completed, then the next unanswered question is selected or incorrectly answered;
- If the concept is completed, we move to the next related concept from the concept map.

A more elaborate strategy can be used to select unanswered questions based on the difficulty level. The two criteria for estimating a learner's success in mastering a concept allow for adaptive strategies, so that the learning path is tailored to the capabilities and knowledge of the students. For example, if a concept is already well known to a student, she will solve a small number of questions correctly, and he will be allowed to progress with a low completeness score due to a high correctness score.

5 Algorithm implementation outlines

In this section, algorithm outlines are given. The top-level code is given and briefly explained.

The main concept generator function is given as Algorithm 2. It turned out to be the most stable concept map generator. The 'rel_shift' option decides concept dependency according to the concept question overlap and weights are set according to it. Only links with weights above threshold w_T are added.

Algorithm 2 Concept map generation from keywords

Require: Set of questions
Ensure: Concept map *conM*

```
kws = get_keywords_from_qs()                          ▷ Generate keywords
conM = get_empty_conM()                               ▷ Get empty concept map
if alg_code == 'min_q' then                           ▷ Minimal overlap approach
    qs_node_attrs = get_min_qs(kws)                   ▷ Get minimal question of the concept.
    sort_conc = sort_concepts()                       ▷ Sort concepts according to qs.
    con_w = set_weights_minq(sort_con)                ▷ Assign weights according to the cocnept order
    conM.add_concepts(con_w)
end if
if alg_code == 'rel_shift' then                       ▷ Relative shift approach
    weighted_edges = []
    qs_node_attrs = get_node_attributes()
    u_v_lims = get_node_limits(qs_node_attrs)         ▷ Estimate concept limits
    for u, v ∈ u_v_lims do
        w_u_v = get_weight(u,v)                       ▷ Estimate weights according to concept limits.
        if w_u_v > w_T then
            weighted_edges.add(u,v,w_u_v)             ▷ Add link if its weihgt is large enough.
        end if
    end for
    conM.weighted_edges(con_w)
end if
```

The algorithm for concept maps merging is based on the exhaustive concept pair comparison where, at the same step, new concepts and new links among them are produced, see Algorithm 3. A merging weight merge_w is used to weigh the relevance of the second concept map conM2 relative to the conM1 (1 is equal relevance, < 1 is less and for > 1 the conM2 is more relevant than conM1).

Algorithm 3 Concept map merging

Require: Concept maps conM1, conM2, merge_w
Ensure: Simplified concept map conM.

```
conM = []
for c in conM1.nodes do                                   ▷ Scann all concept pairs
    for c_a in conM2.nodes do
        if qs(c) ∩ qs(c_a) then                           ▷ Nonempty intersestion
            for nc in nc(c) do                            ▷ For neighbor concepts of c
                for nc_a in nc(c_a) do                    ▷ For neighbor concepts c_a
                    if qa(nc) ∩ qa(nc_a)) then
                        m_ncqs = get_merged_concept_qs(nc, nc_a)   ▷ get merged concepts
                        m_ncqs = get_merged_weight_qs(nc, nc_a)    ▷ Get merged weoghts
                        conM.add_nodes(m_ncqs)
                        conM.add_nodes(m_ncqs)
                    end if
                end for
            end for
        end if
    end for
end for
```

Since automatically generated concepts and their mergings are usually highly fragmented, a simplification is needed, see Algorithm 4. Simplification operations are 1. Concept absorption (when subsets or close to it, remove empty concepts), 2. Link merge (parallel links of links to opposite directions.)

Algorithm 4 Concept map simplification

Require: Concept map conM
Ensure: Simplified concept mam simpl_conM

```
for e in conM.edges do                                    ▷ Remove low weight and parallel edges
    if w(e) < w_T then
        conM.remove_edge(e)
    else if  then
        for f in conM.edges do
            if parallel(e,f) then
                conM.merge(e,f)
            end if
        end for
    end if
end for
for c in conM.nodes do                                    ▷ Remove empty nodes
    if c == [] then
        conM.remove(c)
    end if
end for
simpl_conM = conM
```

The proposed recommender algorithm is based on a graph walk of concept map, learning indicators are optimized inside concepts only. For this concept map walk see Algorithm 5. It is based on a modified BFS walk taking link weights into account. Two options are implemented, i.e. 'Hamilton' (ignores weights) and 'weighted_BFS' providing best results.

Question recommender algorithm (see Algorithm 6) follows the concept map walk defined by Al-

Algorithm 5 Concept map walk

Require: Concept map conM
Ensure: Concept map walk seq_dc

```
c_s = conM.get_starting_c()
if 'Hamilton' then                                              ▷ Use Hamilton walk
    path = nx.algorithms.tournament.hamiltonian_path(conM_S)
    seq_dc = sequence(path)
end if
if 'weighted_BFS' then
    queue = [c_s]
    visited = []
    visited_nexts = []
    while not empty(queue) do                                   ▷ BFS walk
        c = queue.pop()
        cn = argmax(c1 ∈ c.neigbours())                         ▷ Get neighbour with largest weight
        if c_n not in visited then
            queue.push(c_n)
        end if
        curr_c = c.neigbours()
        while not empty(curr_c) do
            cc = curr_c.pop()
            if cc not in visited_nexts then
                visited_nexts.push()
            end if
        end while
        if len(visited) < n then                                ▷ Not all visited?
            cnn = conM.get_next_node()
            queue.append(cnn)
        end if
    end while
    no_prev_c = get_no_prev_c()                                 ▷ Connect sequence
    no_next_c = get_no_next_c()
    for c in no_next_c do
        if len(no_next_c) then
            cn = no_next_c.pop() seq_dc[c] = cn
        end if
    end for
end if
```

gorithm 5 between concepts and maximizes selected learning indicators within concepts by selecting appropriate questions.

Algorithm 6 Question recommender algorithm: recommend next questiob

Require: Concept map conM, the current user profile in output_row
Ensure: Question recommendation q_next and concept conc it belongs to

```
cList = get_conc_seq(conM)
cSeq = get_concepts(cSeq)
update_user_history(output_row)
u_score = calc_score(user_id, users_data, conM)          ▷ Estimate learning indicators
if no user history found then                  ▷ No question found => go to first concept
    conc = cList[0]
    q_next = get_qs(conc)[0]
    return [q_next, conc]
end if
q_next = -1
last_conc = -1
for conc in u_score do                                          ▷ Scan relevant concepts.
    last_conc = conc
    c_compl = get_c_compl(u_score)
    c_succ = get_c_succ(u_score)
    if c_compl < comp_T & c_succ < succ_T then          ▷ Enter concept if not mastered yet
        if q_next < -1 then              ▷ No q in the history q_next = get_untaken_q(conc)
        end if
    end if
end for
if q_next < -1 then                                  ▷ No question found in this concept
    if last_conc in cList then
        q_next = get_qs(next_conc)[0]                ▷ First question of the next concept
        = next_conc(cSeq)
    end if
end if
for conc in u_score do                  ▷ Scan for wrongly anwsered question concepts.
    last_conc = conc
    c_compl = get_c_compl(u_score)
    c_succ = get_c_succ(u_score)
    if c_compl < comp_T & c_succ < succ_T then          ▷ Enter concept if not mastered yet
        if q_next < -1 then              ▷ No q in the history q_next = get_wrong_q(conc)
        end if
    end if
end for
```

6 Conclusion remarks

This report covers the current development of OptLearn algorithm based on concept map generation and BFS-based graph walk. Student group-dependent concept map generation (motivated by [1], [2]) was not stable on our data and was not used in the current version.

Some observations:

1. No automatically generated concept map is directly useful

   => we combine several concept maps: by merging them

   => allows expert intervention: by integration of the teacher's concept map

2. Cold start problem solved: by integration of either the teacher's concept map or trivial concept map (linearly ordered questions)

3. A breakdown of factors determining concept map is supported - from:

    - learning material (implemented, applied)
    - group of students (implemented, not applied)
    - Learner selection (implemented, applied)

The current implementation is stable. Several improvements are possible, mainly dependent on student history data. Time stamps given in output_row are not utilized yet. Optimization of algorithm parameters (see Sec 3) are in order when a larger sample set is available.

References

[1] Shyi-Ming Chen and Shih-Ming Bai. Using data mining techniques to automatically construct concept maps for adaptive learning systems. *Expert Systems with Applications*, 37(6):4496–4503, Jun 2010.

[2] Shyi-Ming Chen and Po-Jui Sue. Constructing concept maps for adaptive learning systems based on data mining techniques. *Expert Systems with Applications*, 40(7):2746–2755, Jun 2013.

[3] Carmen Romero, Moisés Cazorla, and Olga Buzón. Meaningful learning using concept maps as a learning strategy. *Journal of Technology and Science Education*, 7(3):313–332, Jul 2017.

[4] Yuwen Zhou, Changqin Huang, Qintai Hu, Jia Zhu, and Yong Tang. Personalized learning full-path recommendation model based on LSTM neural networks. *Information Sciences*, 444:135–152, May 2018.