The Organization of Large-Scale Repositories of Learning Objects with Directed Hypergraphs

Luigi Laura[1,2(✉)], Umberto Nanni[1,2], and Marco Temperini[1]

[1] Department of Computer, Control, and Management Engineering Antonio Ruberti, Sapienza University, Via Ariosto 25, 00185 Roma, Italy
{laura,nanni,marte}@dis.uniroma1.it
[2] Research Centre for Transport and Logistics (CTL), Sapienza University, Roma, Italy

Abstract. In this paper we focus on the problem of finding personalized learning paths in presence of a large number of available learning components. In particular, we model the relationships holding between learning activities and the related (needed/achieved) competence, by adopting directed hypergraph. We show as the complexity of optimizing learning paths depends dramatically on the adopted metrics; in particular, we prove that finding a learning path with minimum *timespan* can be done in quasi-linear time, whilst finding one with minimum *total effort* (apparently, a very similar problem) is NP-hard. Therefore in some cases, it is possible to use simple and fast algorithms for computing personalized e-learning paths, while in other cases the developers must rely on approximated heuristics, or adequate computational resources. We are implementing this modeling and the related algorithms in the framework provided by the LECOMPS system for personalized e-learning. The final aim is to apply the modeling in large repositories, or in wider web-based e-learning environments.

1 Introduction

The personalization of learning experience is widely recognized as a main success factor in (e-)education. Taking care of personal traits of the learner, and making the learning activities compliant with them is a main issue, in particular, in web-based e-learning. The model of competence-based learning is widely used to render personal learning accomplishments and needs: roughly summarizing, given a target for a study course, basing on the present "state of knowledge" (amount of possessed competence) of the individual learner, a set of learning activities (LAs) can be stated for the learner to undertake, so that only the lacking competences are addressed and the target can be reached with less effort and study time. Provided that assessment means are usable to test the state of knowledge during the course, the set of LAs can also be continuously adapted to the changing learner's model.

In this paper we base on the LMS architecture provided by an existing system for personalized e-learning (LECOMPS). In that system, as in many others,

Y. Cao et al. (Eds.): ICWL 2014 Workshops, LNCS 8699, pp. 23–33, 2014.
DOI: 10.1007/978-3-319-13296-9_3

the basic factors defining the personal student model are competence-based: the knowledge that the individual student possesses, before of the course or at any stated moment during course taking, is used to define and maintain the course structure. Accordingly, competence is also appearing in the definition of the learning objects. In the system numerous learning objects repositories have to be managed, each one possibly amounting to a huge number of items. Moreover learning objects in such repositories are connected by a possibly large quantity of relations (such as the dependency linking one object enabling the taking of another). The repositories can be depicted as graphs, where the amount of nodes (learning objects) and arcs (dependencies) presents the e-learning system with possibly very challenging computational requirements, when automatic construction and continuous adaptation of personalized courses is to be supported.

Directed hypergraph (see, e.g., [10]) can be used to model in a very natural way the relationships holding among learning activities. Several works propose hypergraphs in order to model processes, Petri-nets, or workflow nets (see, e.g. [1,7,14,16]). Interestingly, Sun and Lu [15] propose directed hypergraphs to describe personalized learning processes, together with the relationships among the process itself, the learning component and the learner. They show the effectiveness of using the directed hypergraphs as a robust model for learning processes. Compared to this work, we use directed hypergraphs in a different (somehow, dual) way. Li et al. [12] address analogous problems in the more unstructured context of self-directed and community-based learning, in particular, they consider the difficulty for the learners to locate suitable learning resources.

As observed in [7], in most of today's learning support systems, the structuring of learning activities *is hard-wired and tied to a specific learning domain and system, especially when tightly integrated with the graphical user interface of the system.* Thus, re-usability in different learning systems is restricted.

We focus on the problem of finding learning paths in large repositories (or in a context of web based education) where each learning component is characterized by means of needed and achieved knowledge. In this context we show how directed hypergraph can be used to model the relationships holding among learning activities and skills, and characterize the complexity of finding learning paths according quite natural optimization criteria, showing that finding a learning paths with minimum *timespan* can be done in quasi-linear time, and finding one with minimum *total effort* (apparently, a very similar problem) is NP-hard. These results show as, at least in some cases, it is possible to use fast time algorithms for computing personalized e-learning paths, thus allowing to support large scale e-learning frameworks. In other cases the computational load suggests the adoption of approximated heuristics, or adequate computational resources. In particular, we have started an implementation on the framework provided by the LECOMPS system [9,13] for personalized e-learning.

2 The Lecomps System and the Adopted Learning Model

LECOMPS is a web-based Learning Management System: the functional architecture is shown in Fig. 1. Here we recall the few aspects of LECOMPS that

are related to our proposed directed hypergraph approach, and refer the interested reader to [9,13] for a complete description of the LECOMPS system. The LECOMPS system provides functionalities to support the usual management of learner and teacher users in courses, the authoring activities in repositories of *Learning Components*[1] (LC), and the automated definition of personalized courses, that are then adaptively delivered to learners. Referring to Fig. 1, the set of LCs defined for a given subject matter, collected in a *LC Pool*, is the *Learning Domain* (LD) for the courses on that matter. Each $lc \in LC$ describes a learning activity, in terms of the *effort* needed to take the associated learning material, the *questions* that can be used to test the competence that can be acquired through the learning activity, and the competence denoted in the lc. The competence in a learning component lc is defined by two sets of *Required Knowledge* (lc.RK) and *Acquired Knowledge* ($lc.AK$). The semantics of RK and AK in a lc is of straightforward interpretation: "you need RK to take the lc, and by that you could gain the competence denoted by AK". The knowledge (skills, competence) denoted in the above ways throughout a LC pool, is called the *Knowledge Domain* (KD) of the repositories. In particular, RK and AK in a lc (both subsets of KD) are defined as sets of *Learning Objectives* (LOs). A LO is a predicate $LO(level, keyword, concepts, context)$ where:

- *level* and *keyword* are cognitive features that label the concepts; they are based on the taxonomy of cognitive levels of Bloom [8];
- each *concept* is a designation (identifier) for an ability or skill or knowledge, on which the above mentioned cognitive traits are significant;
- the *context* identifies the learning context of the concept(s).

The labeling of concepts allows to state some derivation rules among LOs: in particular, a competence *cpt* possessed at a certain cognitive level in a given context *ctx* implies the possession of *cpt* at lower levels in *ctx*:

$$LO(3, apply, cpt, cxt) \Longrightarrow LO(2, describe, cpt, cxt)$$

LECOMPS builds personalized courses by synthesizing a set of Learning Components, $\mathcal{C} = \{lc_1, \ldots, lc_n\} \subseteq LD$ (linearizable in a sequence) basing on a statement for the *Target Knowledge* (TK) and an evaluation of the initial state of knowledge of the individual learner (cfr. CS below). The aim of the course is to bridge the gap between the present state of learner's CS and TK:

$$CS^{init}|_{KD}, \bigcup_{i \in [1 \ldots n]} lc_i.AK \vdash TK$$

(meaning that the set of LO comprised in the initial state of CS, together with the LO acquired through the course, entails the target knowledge).

[1] A Learning Component is a SCORM compliant Learning Object, where an instructional content is enriched with specification elements, to make it usable in an automated process of selection [13].

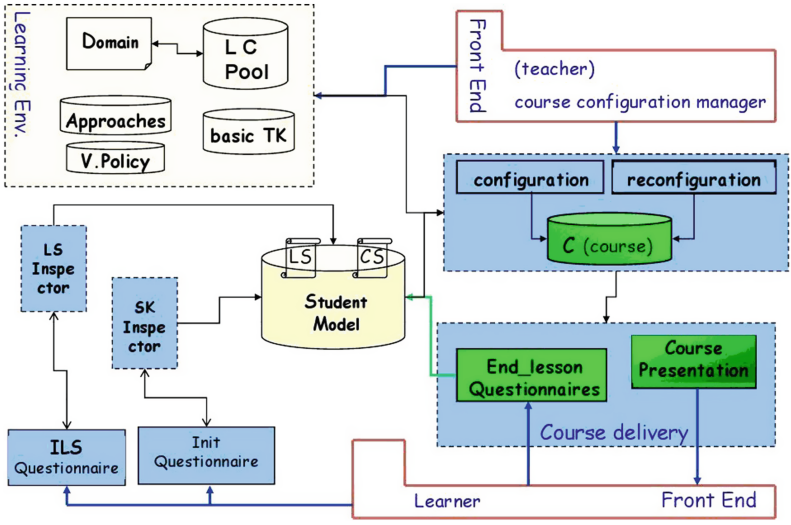

Fig. 1. LECOMPS [13] functional architecture.

3 Modeling Repositories with Directed Hypergraphs

Following the model provided in the previous section, let us suppose to have a set of *LO*s in a given subject matter. Here we will use a simplified version of the *LO* model given in Sect. 2, where we regard a learning objective *lo* as a pair $\langle c, l \rangle$, i.e., a concept c, and a level/degree of competence l; we will denote this as c_l. Moreover we will define a *Repository* $\mathcal{R}$ of Learning Components as a collection $\mathcal{R} = \{lc_1, lc_2, \ldots, lc_m\}$, while each learning component is a 4-tuple $lc_i = \langle C_i, RK_i, AK_i, E_i \rangle$, where (cfr. previous section),

- C_i is the *Learning Content* of the component;
- $RK_i \subseteq KD$ is the *Required Knowledge* in lc_i;
- $AK_i \subseteq KD$ is the *Acquired Knowledge* through lc_i;
- E_i is the *Effort* (quantitative measure of global effort endured to take C_i).

We will denote as $LO(\mathcal{R})$ and $\mathcal{C}(\mathcal{R}) = \mathcal{C}(LO(\mathcal{R}))$, resp., the set of learning objectives used in $\mathcal{R}$ and the collection of concepts used to define the learning objectives in $LO(\mathcal{R})$.

Directed Hypergraphs and Hyperpaths. Directed hypergraphs are a generalization of directed graphs which have been used to model properties of structures in a variety of contexts (see, e.g., [2,6,10]). A *directed hypergraph* $\mathcal{H}$ is a pair $\langle N, H \rangle$, where N is a set of *nodes* and $H \subseteq \mathcal{P}^+(N) \times \mathcal{P}^+(N)$ is a set of *hyperarcs*, where $\mathcal{P}^+(N)$ is the collection of nonempty subsets of N. Each hyperarc is an ordered pair $h = (S, T)$, where both the *source set* S (or *head*), and the *target set* T (or *tail*) are arbitrary nonempty sets of nodes: $S, T \subseteq N$.

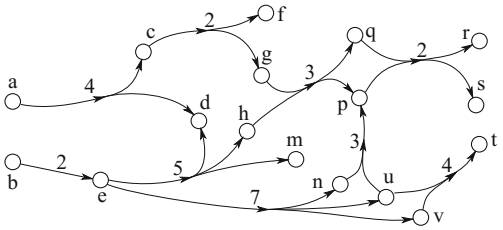

Fig. 2. An example of weighted directed hypergraph.

A *weighted directed hypergraph* $\mathcal{H}_W$ is a triple $\langle N, H; w\rangle$, where $\langle N, H\rangle$ is a directed hypergraph and each hyperarc $\langle S, T\rangle \in H$ is associated to a real value $w_{\langle S,T\rangle} \in \Re$ called the *weight* of the hyperarc. An example of weighted directed hypergraph is given in Fig. 2.

The hyperpaths that we are interested to will never use twice the same hyperarc. Based on this observation, for the purposes of this paper, we can define hyperpaths as follows. Let $\mathcal{H} = \langle N, H\rangle$ be a directed hypergraph, $X \subseteq N$ be a non-empty subset of nodes, and y be a node in N. A *hyperpath* from X to y in $\mathcal{H}$ is a minimal set of hyperarcs such that one of the following conditions holds:

(a) $y \in X$ (*extended reflexivity*); in this case, the hyperpath $h(X, y)$ is the empty set;
(b) there is a hyperarc $(Z, y) \in H$ and hyperpaths from X to each node $z_i \in Z$ (*extended transitivity*); in this case, the hyperpath $h(X, y)$ includes the hyperarc (Z, y) and all the hyperpaths $h(X, z_i)$.

If there exists a hyperpath from X to y, then we say that vertex y is *reachable* from X, or $X \rightsquigarrow \{y\}$. We can generalize this definition to a hyperpath from a source set $X \subseteq N$ to a target *set* $Y \subseteq N$ as a union of hyperpaths $h(X, Y) = \bigcup_{y_i \in Y} h(X, y_i)$. If there exists a hyperpath from X to Y we say that the vertices in Y are all *reachable* from X, or $X \rightsquigarrow Y$.

In Fig. 2, we can see that $\{bg\} \rightsquigarrow \{q\}$, in fact there is the hyperpath $h(bg, q) = \{(b, e), (e, dhm), (gh, pq)\}$, but $\{b\} \not\rightsquigarrow \{q\}$, since the only hyperarc entering node q is (gh, pq), but there is no way to reach g from b. On the other side, $\{ab\} \rightsquigarrow N$, that is, from the source set $\{a, b\}$ we can reach any node in the hypergraph.

Hyperpaths metrics. There are many possible ways to measure a directed hyperpath. A quite natural metric is to measure the `size` of the hyperpath, which can be defined as the sum of the weights of all the hyperarcs, that is:

$$\texttt{size}(h(X, y)) = \sum_{(S,t)\in h(X,y)} w(S, t)$$

In particular, the `size` of the empty hyperpath is zero.

Another possible metrics consist in the `rank` of the hyperpath, defined as the maximum consecutive chaining of the weights. In particular, the `rank` of an

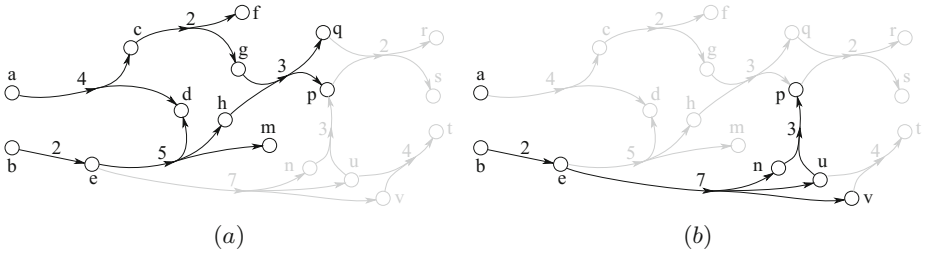

Fig. 3. Here we look for an "optimal" hyperpath $h(ab, p)$ for the hypergraph in Fig. 2. The hyperpath shown in (a) has size=16 and rank=10; the one in (b) has size=12 and rank=12. Hence, the choice is not unique, since it is based on the criterion of optimality.

empty hyperpath is zero, whilst the rank of a hyperpath defined by transitivity, that is $h(X, y) = \{(Z, y)\} \cup \bigcup_{z_i \in Z} h(X, z_i)$ is defined as follows:

$$\texttt{rank}(h(X, y)) = w(Z, y) + \max_{z_i \in Z}\{\texttt{rank}(h(X, z_i))\}$$

Examples of "optimal" hyperpaths according to the above metrics are in Fig. 3.

A Hypergraph for a Repository of Learning Components. Let us consider a Repository $\mathcal{R} = \{lc_1, lc_2, \ldots, lc_m\}$ of Learning Components, with the corresponding set of Learning Objectives $LO(\mathcal{R})$ and the set of Concepts $\mathcal{C}(\mathcal{R})$ used in LO. We can define a corresponding *Learning Hypergraph* $\mathcal{H}_{\mathcal{R}}$, according the following rules:

- the set of nodes coincides with the set of Learning Objectives: $N \equiv LO$; we recall that each learning objective lo with concept c and level l is denoted simply as c_l;
- for each learning component $lc_i = \langle C_i, RK_i, AK_i, E_i \rangle \in \mathcal{R}$ we introduce the hyperarc (RK_i, AK_i) whose weight is equal to the effort E_i;
- for each concept $C \in \mathcal{C}(R)$ and for each i, j such that $1 \leq j < i \leq k$, we introduce "implicit" hyperarcs (c_i, c_j) with weight 0.

The implicit arcs do not correspond to actual Learning Contents, but implement the assumption (stated in Sect. 2) that a competence possessed at a certain cognitive level implies the possession of competencies at lower levels in the same context (with zero effort):

$$c_i \Longrightarrow c_j \quad \text{for each level } j < i.$$

Note that an ontology, established among the concepts in KD, might introduce analogous implications, due to concepts which are conceptually related. As an example, if a concept c' *subsumes* a concept c'' (e.g., if c'' is a special case of concept c'), then one might assume that, if a learner reaches an educational goal c'_i (concept c' at level i), then (s)he reaches the educational goal c''_i as well (i.e., $c'_i \Longrightarrow c''_i$, for each level i). If this is the case, we can introduce an implicit arc from c'_i to c''_i for each $i = \{1, 2, \ldots, k\}$ with effort 0.

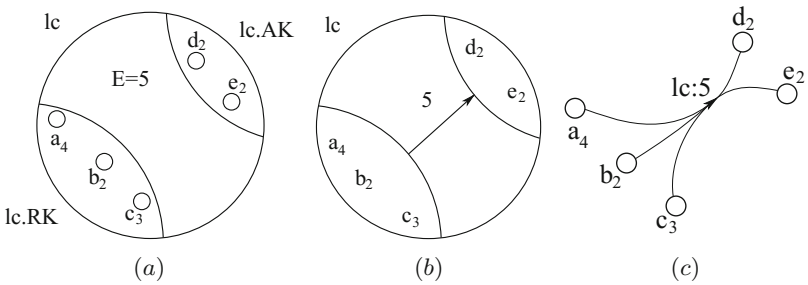

Fig. 4. Representation of a single learning component lc: (a) graphical representation of lc, characterized by a Required Knowledge $\{a_4, b_2, c_3\}$, an Acquired Knowledge $\{d_2, e_2\}$, and an Effort $E = 5$; (b) a simplified representation of the same lc; (c) the same lc represented as a directed hyperarc from $lc.RK$ to $lc.AK$.

Modeling Learning Paths with Hypergraphs. Adopting directed hypergraphs allows us to exploit the algorithms conceived for this model to find *Learning Paths* with certain characteristics from the current cognitive state of the learner to any possible target knowledge.

Let us consider a repository $\mathcal{R}$ of learning contents and a learner in a given state of knowledge $CS \subseteq KD$. A *(feasible) Learning Path* is a sequence $LP = \langle lc_1, lc_2 \ldots, lc_m \rangle$ such that each $lc_i \in LP$ has a required knowledge $RK_i \subseteq CS \cup AK_1 \cup AK_2 \cup \cdots \cup AK_{i-1}$, for $i = 1, 2, \ldots, m$. Intuitively, a feasible learning path is such that each learning component lc_i has a required knowledge that is part of the *current* cognitive state of the learner, which includes the initial cognitive state plus the acquired knowledge of the learning components *before* lc_i in the sequence. In the following, when no ambiguity arises, we refer only to feasible Learning Paths.

Let us consider a repository of Learning Components $\mathcal{R}$, a learner in a cognitive state $CS \subseteq KD$, and a target knowledge $TK \subseteq KD$. If we consider the Learning Hyperpath $\mathcal{H}_\mathcal{R}$, the following properties hold:

1. There exists a learning path LP from CS to TK in $\mathcal{R}$ if and only if there exists a hyperpath $CS \rightsquigarrow TK$ in $\mathcal{H}_\mathcal{R}$; furthermore, any topological order of the hyperpath $h(CS, TK)$ is a feasible learning path LP;
2. A learning path with minimum *total effort* (computed as the sum of the efforts of all the involved LCs) corresponds to a hyperpath with minimum `size`, and vice-versa;
3. If we suppose that more learning components can be taken in parallel, a learning path with minimum *timespan* corresponds to a hyperpath with minimum `rank`, and vice-versa.

Of course, in a realistic setting, we would impose a bound on the effort of parallel learning activities: this can handled by defining optimization criteria with constraints.

An example. In order to make more intuitive the use of the proposed model, we provide a small example of a repository of LCs modeled by a directed hypergraph. In Fig. 4 we can see the graphical representation of a single learning content lc; in particular we see the same lc depicted in three distinct ways: a general representation, a simplified one, and the corresponding hyperarc. A sample repository $\mathcal{R}$ is given in Fig. 5 which shows, respectively, the learning components included in this repository, and the corresponding Learning hypergraph $\mathcal{H}_{\mathcal{R}}$. Beside the learning components, this hypergraph includes dashed arcs representing the implicit arcs at zero cost. Note that the third Learning Content, C, represents a course improving skill a from level 3 to level 5.

Suppose that a given learner with initial cognitive state $CS = \{a_4\}$ must reach the Target Knowledge $\{f_2, g_2\}$. This is possible by a learning path $\langle E, F\rangle$, with timespan 6 and total effort 6 (see Fig. 6(a)). On the other side, a less advanced learner with cognitive state $CS = \{a_3\}$ may follow $LP = \langle A, B, D, F\rangle$ (Fig. 6(b)), with a timespan 7 and effort 9. In this case, a learning path with smaller effort would be $\langle C, E, F\rangle$ (Fig. 6(c)), with effort 8, but timespan 7.

The computational cost of computing Learning Paths. The size of the description of a repository $\mathcal{R}$ corresponds to the size of the Learning Hypergraph. For each learning component lc_i we need to represent the required knowledge RK_i and the acquired knowledge AK_i, possibly sharing this representation among several learning components. The space requirement to represent n learning components is $O(n+s)$, with $s = \sum_i |S_i|$, where either $S_i = AK_i$ or $S_i = RK_i$ for some lc_i. In the model described above, given a repository $\mathcal{R}$ (with a description of size $|\mathcal{R}|$), an initial cognitive state CS, and a target knowledge TK, we can prove the following results:

1. finding a feasible Learning Path (or checking its absence) can be solved in linear time, that is $O(|\mathcal{R}|)$; the available algorithms are *incremental*, that is, the solution can be updated for small changes of the repository (this corresponds to finding any hyperpath from CS to TK [4]);
2. finding a Learning Path with minimum timespan can be solved in time $O(|\mathcal{R}| \cdot \log |\mathcal{R}|)$ (as finding a hyperpath with minimum rank [5]);
3. finding the hyperpath with minimum total effort is NP-hard (this is the complexity of finding a hyperpath with minimum size [3]).

We claim that the first two results can be stated (with the same time complexity), even if we consider the problem of finding any learning path (or a hyperpath with minimum timespan) with *filtering*, e.g., involving skill levels not larger than k. The third result is a lower-bound for the problem of finding a learning path with minimum total effort, within the considered model. Of course, there are approximated approaches to this problem (see, e.g., [11]), due to its reducibility to the well known *set covering* problem. A classification of optimization criteria for directed hyperpaths and the hardness of the related problems is given in [3].

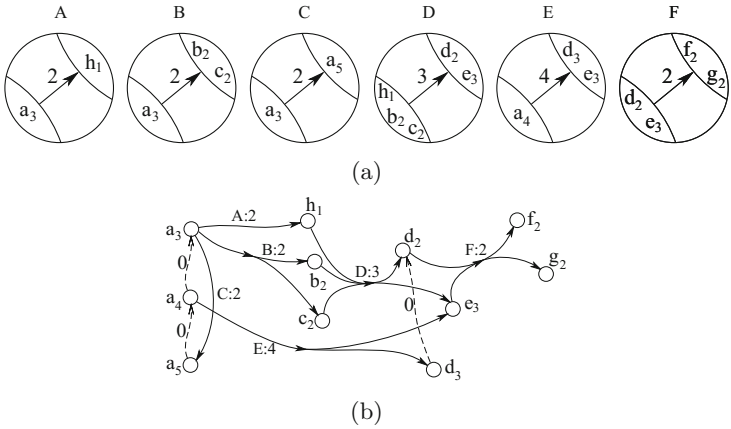

Fig. 5. (*a*) A repository of Learning Contents $\mathcal{R}$ with six learning components; (*b*) the corresponding learning hypergraph $HH_{\mathcal{R}}$ including a hyperarc for each learning component plus three implicit arcs with zero cost: these correspond to an implicit containment (for any concept c, an educational objective at level i implies an educational objective at a lower level, i.e., $c_i \Longrightarrow c_{i-1}$).

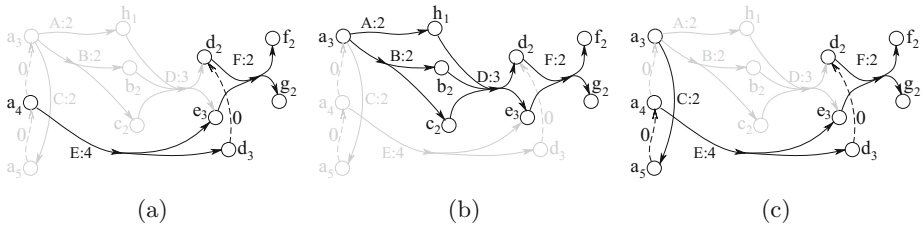

Fig. 6. Referring to the repository in Fig. 5, given a Target Knowledge $\{f_2, g_2\}$:
(*a*) a Learning Path from a Cognitive State $CS = \{a_4\}$ with effort=6 and timespan=6;
(*b*) a Learning Path from a Cognitive State $CS = \{a_3\}$ with timespan=7 (and effort=9);
(*c*) a Learning Path from $CS = \{a_3\}$ with effort=8 (and timespan=8).

4 Conclusions

Populating a repository (pool) with a huge number of learning objects and related connections might derive in long waiting times for the production of courses, as well as in long and even less sustainable delay in adaptation. We have shown some promising ways to find customized learning paths within repositories of learning objects through the use of directed hypergraphs. This opens the possibility of using algorithms and heuristics devised on such data structures, to ease the representational burden of large pools and the computational greed of certain operations, such as course content selection, learning object sequencing, and searching for optimal learning paths. These operations are notoriously of the highest computational complexity, and are usually disposed of by confiding in

the limited cardinality of the learning objects repository at hand. With the presented approach we hope to devise better solutions to the problem, by improving the possibility of exploiting decades of algorithmic results developed in the area of directed hypergraphs for the challenging goal of a sustainable management of large repositories of learning objects. Further algorithmic problems are to be considered, such as the optimization of learning paths with multiple criteria.

References

1. Alimonti, P., Feuerstein, E., Laura, L., Nanni, U.: Linear time analysis of properties of conflict-free and general petri nets. Theor. Comput. Sci. **412**(4–5), 320–338 (2011)
2. Ausiello, G.: Directed hypergraphs: data structures and applications. In: Dauchet, M., Nivat, M. (eds.) CAAP 1988. LNCS, vol. 299, pp. 295–303. Springer, Heidelberg (1988)
3. Ausiello, G., Italiano, G.F., Laura, L., Nanni, U., Sarracco, F.: Structure theorems for optimum hyperpaths in directed hypergraphs. In: Mahjoub, A.R., Markakis, V., Milis, I., Paschos, V.T. (eds.) ISCO 2012. LNCS, vol. 7422, pp. 1–14. Springer, Heidelberg (2012)
4. Ausiello, G., Italiano, G.F., Nanni, U.: Dynamic maintenance of directed hypergraphs. Theor. Comput. Sci. **72**(2–3), 97–117 (1990)
5. Ausiello, G., Italiano, G.F., Nanni, U.: Hypergraph traversal revisited: cost measures and dynamic algorithms. In: Brim, L., Gruska, J., Zlatuška, J. (eds.) MFCS 1998. LNCS, vol. 1450, pp. 1–16. Springer, Heidelberg (1998)
6. Berge, C.: Graphs and Hypergraphs. Elsevier, Amsterdam (1973)
7. Bergenthum, R., Desel, J., Harrer, A., Mauser, S.: Modeling and mining of learnflows. In: Jensen, K., Donatelli, S., Kleijn, J. (eds.) ToPNoC V. LNCS, vol. 6900, pp. 22–50. Springer, Heidelberg (2012)
8. Bloom, B.E.: Taxonomy of Educational Objectives. D.McKay Company Inc., New York (1964)
9. De Marsico, M., Sterbini, A., Temperini, M.: A framework to support social-collaborative personalized e-learning. In: Kurosu, M. (ed.) HCII/HCI 2013, Part II. LNCS, vol. 8005, pp. 351–360. Springer, Heidelberg (2013)
10. Gallo, G., Longo, G., Nguyen, S., Pallottino, S.: Directed hypergraphs and applications. Discrete Appl. Math. **42**, 177–201 (1993)
11. Gallo, G., Pallottino, S.: Hypergraph models and algorithms for the assembly problem. Technical report 06/92, Dip. di Informatica, University of Pisa, Italy, Corso Italia 40, I-56125 Pisa, Italy (1992)
12. Li, H., Hasegawa, S., Kashihara, A.: A resource organization system for self-directed & community-based learning with a case study. In: Wong, L.-H. et al. (eds.) I. A.-P. S. for Computers in Education, Proceedings of the 21st International Conference on Computers in Education - ICCE 2013, pp. 329–338 (2013)
13. Limongelli, C., Sciarrone, F., Temperini, M., Vaste, G.: The lecomps5 framework for personalized web-based learning: a teacher's satisfaction perspective. Comput. Human Behav. **27**(4), 1285–1466 (2011)
14. Liu, X.-Q., Wu, M., Chen, J.-X.: Knowledge aggregation and navigation high-level petri nets-based in e-learning. In: International Conference on Machine Learning and Cybernetics, vol. 1, pp. 420–425 (2002)

15. Sun, X., Lu, Y.: Directed-hypergraph based personalized e-learning process and resource optimization. In: 2012 Fourth International Conference on Digital Home (ICDH), pp. 171–178, (Nov 2012)
16. Xuedong, S., Feng, Z.: Unified and integrated e-learning modeling supporting dynamic learning process optimization. In: Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 4, pp. 2137–2141 (July 2011)