An Approach Towards Designing Problem Networks in Serious Games

A. Parakh
School of Interdisciplinary Informatics
University of Nebraska Omaha
Omaha, NE
aparakh@unomaha.edu

P. Chundi
Computer Science Department
University of Nebraska Omaha
Omaha, NE
pchundi@unomaha.edu

M. Subramaniam
Computer Science Department
University of Nebraska Omaha
Omaha, NE
msubramaniam@unomaha.edu

***Abstract*—Integrating of subject matter onto serious games is an important problem that has been shown to impact the learning potential of serious games. A novel approach, inspired by peer-to-peer (P2P) networks, towards designing and deploying a series of problems in game scenarios is described. Given a set of problems involving a set of concepts the proposed approach automatically generates a problem network graph akin to P2P network that can then traversed by a player to collect all the concepts that are necessary to learn a topic of interest. A network traversal algorithm is described, which identifies the relevant problems and produces an efficient route through the network for learning the topic. We also describe an algorithm for mapping the problem network graph onto a game scenario by identifying groups of problems that can be placed in a single location of the game like the level of a building, arcade, or a room, physical barriers that separate, and the conditions for passing through these barriers. The proposed approach has been validated through a quantum cryptography game *QuaSim* and has been played by over 100 students to learn quantum cryptography basics and cryptography protocols.**

***Index Terms*—Serious games, game design, peer-peer networks, quantum cryptography, quantum computing**

I. Introduction

Serious games have achieved impressive progress in aiding engaged learning in diverse areas involving complex and subtle concepts. The field of cybersecurity has specially benefited from serious games where often complex cybersecurity concepts can be modeled as competitions or attack/defend scenarios [1]–[3]. The integration of subject matter in serious games have enabled players to master concepts in a hands-on manner with a high degree of engagement. However, the design of instructional activities (problems) and their layout in a game scenario has usually been somewhat adhoc relying mostly on feedback from players. Not much attention has been devoted to systematically integrating problems onto game scenarios.

Systematic design of such game scenarios is a challenging problem since problems involving a reasonable number of coherent concepts must be designed and they must be integrated so that players can learn a topic of interest by solving a reasonable number of problems in the game. Further, the problems must be physically distributed in a game scenario so that the players are able to discover problems easily without getting frustrated by having to attempt problems that are unrelated to the topic that they are trying to learn.

In this paper, we propose a systematic approach to design, order and distribute problems in a game scenario that enables players to achieve proficiency in a topic by efficiently locating and solving a series of problems in the scenario. Our proposed approach is inspired from the design of peer-to-peer (P2P) networks whose nodes correspond to the problems and there is an edge between two problems if the player can move from one problem to another in the scenario. The concepts underlying a problem are mapped to the resources available at a node and a player can navigate the network, discovering nodes and collecting resources at each node until sufficient resources are collected to achieve proficiency in a topic of interest.

The proposed approach analyzes the given dependencies among a set of concepts to automatically generate a P2P network of problems. Groups of problems in the network are identified and mapped to locations in a game scenario such as levels in a building, arcade or a room, and the physical barriers in a game scenario such as walkways, stairs, or elevators separating these groups are also identified. These serve as design specifications to designers who can then implement problems in game scenarios optimized for layout and usability.

Given a game scenario with a problem network, players are expected to traverse this network solving problems that involve concepts that are necessary to learn a certain topic. Since the problem network is designed to allow learning of multiple topics, it allows players a lot of flexibility in moving between problems. Many of these problems may not be relevant to the topic being learnt and hence attempting such problems is not very fruitful to the task at hand and may also cause unwanted frustration in players. We describe an automatic network traversal algorithm to address this problem. Given a topic of interest, our algorithm identifies the learning gain at each node in the problem network and chooses a route based on maximizing the learning gain among viable neighbor nodes at each point in the route. The algorithm enables players to learn a topic by minimizing their attempts at solving problems to the topic.

This research has been supported by NSF award #1623380

The proposed system has been used to design problems in the serious game *QuaSim* [4]–[6], a gamified virtual education platform. QuaSim has been played by over 100 undergraduate and graduate students who successfully used the implemented problem network and the traversal algorithm to learn several aspects of quantum computing and quantum cryptography. Our results about user learning and experiences in using QuaSim are reported in [4]. In this paper, we focus on the approach used to design and implement the problem network.

The rest of the paper is organized as follows. Section II discusses related work. Section III defines terms and describes the algorithms to generate a P2P problem network and mapping of this network to game scenarios. In Section IV, we show how the proposed algorithm are applied to generate game scenarios with problems in QuaSim and how topics can be learnt efficiently by using our network traversal algorithm. Section V concludes the paper.

II. Related Work

Game-based learning plays a crucial role in cybersecurity education where games such as capture the flag or the red and blue teams of cyber attackers and defenders are commonly used to learn security constructs in a hands-on manner [1]–[3].

A. *Quantum Education Games*

The inherent interdisciplinary nature of quantum computing and quantum cryptography involving concepts from quantum physics, computer science, mathematics and cybersecurity pose a formidable challenge for students wanting to learn these areas. In recent years, there have been some novel attempts to develop games that teach the laws of quantum physics and quantum mechanics to students [7]–[15]. Other approaches include interactive tutorials [16] for teaching non-intuitive quantum physical concepts. Most of the games that have been developed are aimed at teaching quantum mechanics to Physics students or to K-12 group of students. Further, these games are traditional two dimensional games ranging from Mario type simulations to puzzles that require moving game board pieces. The amount of instructional support is quite limited in these rudimentary games and often solely focus on quantum principles and not on their application to cybersecurity or quantum computing. Further, the games do not adapt to observed player behavior in order to maximize learning and engagement.

QuaSim is, however, an adaptive serious game that is focused on teaching quantum mechanical principles needed for quantum cryptography and the implementation of quantum key distribution protocols [4]–[6]. QuaSim consists of a variety of instructional tools such as videos, quizzes, audio narrations, gaming missions, chats, and web browser within a media rich 3D virtual environment built on Unreal Engine 4.

B. *Serious Games for Learning*

Advances in game-based learning technologies have led to tremendous increase in interest in serious games that combine pedagogical techniques and video games [17]–[21]. Arnab et al. [17] propose a model for integrating educational and gaming elements into serious games along with a criteria for designing and evaluating learning objectives in serious games. A methodology to integrate subject matter and game design principles using game mechanics, conditions for gamification based on pedagogical levels, and implementation and evaluation of the gamified application is discussed in [18]. The importance game design plays in learning of subject matter objectives and its effect on the gaming experience is discussed in [21]. Slimani et al. [22] classify video game elements based on their suitability for serious games and describe how to design these elements.

Bayesian Knowledge Tracing (BKT) is a popular approach to modeling and predicting student performance in an Intelligent Tutoring System (ITS) [23]. It builds and updates a model for the knowledge gained by a student as they apply the skills they've acquired previously (through readings and lectures) while solving problems [24]. Although related, this is different from the proposed game-based approach where completing challenges in the game itself is the means to acquire skills. Therefore, unlike BKT, there is no prior learned/unlearned state for a skill that is fed into the system. BKT may, however, be used as a alternative navigation guidance algorithm to the one described here.

Researches have studied the impact that serious games have on the learning of subject matter and student's engagement in the subject matter [19], [25]–[28]. For example, Hamari et al. [19] have defined the concepts of engagement, flow and immersion and examined the effect these have on the learning of subject matter in a game-based virtual learning environment. Smith et al. [26] highlight the key challenges in designing and developing games for STEM discipline. Sung et al. [28] studied the effect of student behavior on learning in a game-based learning activity for elementary students. They conclude that the high performers showed deep thinking and reflective behavioral patterns when compared to low performing students.

Knowledge and learning space theories [29] provide a framework to represent and assess a body of knowledge acquired by a student. These theories have been extensively applied in many universities and schools because they are able to predict what a student is ready to learn. However, there is still a need to combine the knowledge elements with game design because research has indicated that game design and proper integration of subject matter with gaming elements is crucial to maximize learning. In the following sections, we describe a systematic approach to distribute challenges in the game and manage the location and player navigation through user feedback.

III. Problem Network Model

In this section, we describe a simple procedure to create a model network of problems given a concept hierarchy graph. The problem network model is an undirected graph whose nodes are problems. A navigation procedure over this network and the mapping of the network and the procedure onto a serious game are subsequently described.

A. Preliminaries

Let C be the set of all the **concepts** and $c_i, i = 1$ to m denote the individual concepts. A **concept hierarchy**, CH is a directed acyclic graph, whose nodes are individual concepts and an edge (c_u, c_v) indicates that to learn concept c_v, a player must learn the concept c_u. Concept c_v **depends** on concept c_u, if edge (c_u, c_v) belongs to CH. Otherwise, concepts c_u and c_v are **independent**.

A **topic** T_j is a subset (non-empty) of the concept set C. A **learning problem**, P_i, consists of k mutually independent, distinct concepts. A game comprises of N problems, $P_1, P_2, \ldots, P_N$, each consisting of a unique combination of concepts.

Designing individual problems entails significant domain expertise since coherent concepts must be identified and meaningful problems involving these concepts must be formulated. Further, certain concepts are best learned together in the context of others whereas others are best learned in isolation. Also, designing problems with too few concepts may lead to long play sessions whereas loading up problems with concepts may lead to overly difficult problems. In view of these challenges, we assume that a set of N problems and the underlying concepts are pre-designed by domain experts and available. Without loss of generality, we assume that each problem uses the same number of k underlying concepts. We choose k to be the maximum number of the concepts used over all the problems and problems using lesser than k concepts are extended using a default concept $true$ (value 1) that every player is assumed to be proficient in.

We define three additional terms that are used in the our navigation procedure that allows players to achieve proficiency in a topic by solving a series of related problems.

Concept Tuple (Ct)**:** It is a tuple of size $|C|$. This keeps track of all the acquired concepts at any point during the game (akin to player status). A 1 in position i denotes that the concept c_i has been acquired and a 0 indicates otherwise. Initial state of the concept tuple is all 0s.

Target Tuple (Tp)**:** This tuple is also of size $|C|$ and with concepts listed in the same order as the concept tuple. A 1 in the i^{th} position indicates that the concept must be acquired in order to learn topic T_j. Remaining positions have 0s in them.

Node Tuple (Nt)**:** This tuple lists concepts in the same order as the other two tuples above with 1s in the positions that the current problem/node covers.

Learning Gain (Lg)**:** It is defined as Lg= HammingWeight[$(Nt$ OR $Ct)$ AND Tp]. We want to maximize the learning gain at every hop.

We assume that all the tuples consist of binary values and, therefore, either the player acquires a concept of not. However, more general tuples with non-binary values may also be considered.

B. Designing Problem Network Model

To design problem networks in game scenarios, we introduce **game nodes** denoting game artifacts like a walkway, stairs, or an elevators etc. A game node is a connection element (like a switch) between problems. A game node becomes **enabled** by solving one or more problems. A game node when enabled allows a player to access additional learning problems to which it connects.

Given concept hierarchy CH over the concept set C and N problems each involving k concepts from C, a problem network model, PN is an undirected graph whose nodes are the N input problems along with one or more game nodes. The labeled edges of PN are either **navigation edges** or **game edges**. Problem nodes are connected using navigation edges. Game nodes and problem nodes are connected using either a game edge or a navigation edge but not both. Game nodes are not connected to other game nodes.

A navigation edge between two problem nodes allows a player to move between these problems without any restrictions. A game edge allows a player to *pass through* only if it is enabled. A game node is enabled only when the player has solved all the problem nodes connected to the game node using a game edge. Hence a player can move between two problem nodes connected through a game node only when the latter has been enabled.

For example, given four problems P_1 = $\{c_1\}$, P_2 = $\{c_2\}$, P_3 = $\{c_3\}$, and P_4 = $\{c_4\}$, some of the possible concept hierarchies and their corresponding problem network models are depicted in Figure 1. The navigation edges are shown as dashed lines whereas the game edges are solid lines.

In the first pair of graphs at the top where the concepts are mutually independent (indicated by the absence of edges in CH, depicted on the left). In this case, a player can solve the problems in any order (indicated by the fully interconnected PN on the right). For the middle pair of graphs, the serial dependencies among the concepts in CH (shown on the left) result in a game nodes $g1$-$g3$ connecting the problems in PN (shown on the right) enforce the concept dependencies.

In the lower pair of graphs, the dependency of C_2 on concepts C_1 and C_3 results in the two game edges between the problems P_1 and P_3 and the game node $g2$ that is connected to problem P_2 through a navigation edge. The game node $g2$ is enabled only when the problems P_1 and P_3 are solved, and then allows the player to move to problem P_2. Note that there is no path between problems P_2 and P_4 in PN, even though the concepts C_2 and C_4 are independent. This is because C_4 depends on C_3 and hence a player can attempt problem P_4 only after P_3. Note however that we can add a navigation edge from P_2 to g_3 indicating that a player can reach g_3 after solving P_2 but they can progress further only if they have solved P_3. These edges can then be used by designers to optimize the design layouts of problems in a game scenario as discussed later in Section 3.

The following algorithm constructs the problem network graph given concept hierarchy CH and a set of problems P = $\{P_1, \cdots, P_n\}$ as inputs. We assume that all the concepts

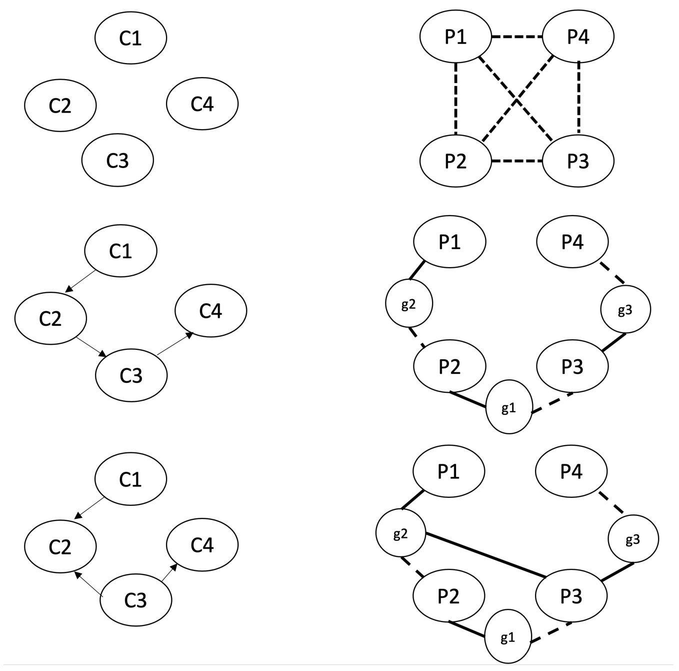


Fig. 1. Concept Hierarchy and Problem Network Model Graphs

in P appear in CH, and that the problems involve a unique combination of concepts that are mutually independent.

A problem P_y belongs to the **dependency set** of a problem P_x if some concept C_x in P_x depends on a concept C_y in P_y (there is an edge from C_y to C_x in CH). If P_y belongs to $D(P_x)$ then P_x can be attempted only after solving problem P_y. The dependency set, $D(P_x)$ of problem P_x is the set of all the problems on which the problem P_x depends. We start with all problems as nodes in PN and an empty edge set.

Algorithm: Problem Network Construction

1) Compute the dependency set, $D(P_x)$ for each problem P_x using the concept hierarchy CH.
2) Add a navigation edge between two problem nodes P_x, P_y in $D(P_x) = D(P_y)$.
3) Given problem P_x, for each problem P_y in $D(P_x)$, if there is a navigation edge between some game node g and P_x, then add a game edge between g and P_y. Otherwise, add a new game node g to PN and add a game edge between P_y and g and a navigation edge between g and P_x.

The dependency levels in the concept hierarchy CH determine the connectivity and the number of hops (problem that need to be solved) between any two problems in the problem network. In the worst case, the number of hops between two problems P_x and P_y equals the maximum path length between the concepts underlying P_x and P_y in CH. Further, a player can learn a topic T_i using at most M hops where M is the longest distance among two problems including the concepts in T_i. The upper bound on the number of problems that a player needs to solve to learn a group of topics is given by the sum of problems required to learn topic and this occurs when the topics are mutually disjoint.

C. Mapping a Problem Network on a Game Scenario

The problem network is mapped on a game scenario by identifying groups of problems that can be placed on a single location in the game like a room, corridor, or a single level where a player can freely move between the problems attempting one after another. The groups of problems are separated from each other by barriers such as doors, elevators, walkways, and stairs and these barriers can be crossed only after groups in the problems can be solved.

Let $E(g)$ be the **enabling set** of problem nodes that need to be solved to enable the game node g. The following mapping procedure takes the problem network graph and the dependency sets of the N problems, as its two inputs. Its two outputs are: i) a partition of problems and ii) the enabling sets for each game node in PN.

Algorithm: Problem Network Mapping

1) Construct the graph $PNminusG$ by deleting all game nodes and the associated edges from PN. Identify the maximal cliques $CL_1, \cdots, CL_v$ including trivial cliques (single nodes) in $PNminusG$. Collect the problem nodes in each clique and add them to the output partition.
2) For each game node, g and edge, (g, P_y), in PN, add the problem node P_y to $E(g)$ if P_N does not have an edge (g, P_x) and P_y belongs to $D(P_x)$, i.e., P_x depends on P_y and P_x is necessary to enable the game node g.

The first step identifies maximal groups of problems that a player can attempt without any restriction. The second step identifies the conditions under which the game nodes (barriers) separating these groups of problems are enabled. It ensures that the enabling set for each game node includes mutually independent problem nodes only. If a problem P_y belongs to $D(P_x)$, and P_x is already included in the enabling set then problem P_y need not be included in that enabling set since a player can attempt problem P_x only after solving P_y.

The next step after designing the graph is to develop a navigation procedure to collect all the resources (concepts) pertaining to a topic. The navigation procedure we use is intended to find the shortest possible path, from any given problem in the game, such that all concepts related to the topic(s) of interest are covered. Shortest possible path is the one that leads the student to solving fewest number of problems while achieving proficiency in each of the concepts encoded into a given problem. Section III-D discusses the procedure for routing procedure for traversing the graph.

D. Network Traversal

The network traversal algorithm simply maximizes the metric learning gain (Lg) at every step. The failure to solve a problem in the game is akin to a node failure in P2P networks and the corresponding resources (concepts) are lost. The network traversal algorithm is robust to such failures. We have assumed that all concepts are replicated on at least r problems so that a student still has a chance to learn those concepts in case they fail to solve a particular problem.

Algorithm: Network Traversal

We define the initial problem set P_{init} as the set of problems whose concepts don't have any dependencies or incoming edges in the concept hierarchy CH. Every problem P_x is associated with a node tuple Nt_x. We traverse the graph as follows,

1) We find a problem P_f in P_{init} that maximizes the learning gain Lg as follows,
    a) Compute the node tuples Nt_x associated with every problem in P_{init}.
    b) Compute the candidate concept tuples for each problem as $Ct_p = Nt_x$ OR Ct.
    c) Choose a node in P_{init} that maximizes the learning gain Lg computed on the candidate concept tuple and the target tuple.
    d) Update $Ct = Ct_p$ if the problem was solved successfully.
2) After identifying and solving P_f, we process the neighbors of a node starting with P_f in G as follows,
    a) For each neighbor $P_{x_1}, P_{x_2}, \ldots, P_{x_m}$ we check if any of the problems is in the initial problem set P_{init}.
    b) For each neighbor that is in P_{init} we compute the learning gain Lg and choose the one that maximizes Lg.
    c) Repeat the above two steps until no more neighbors in P_{init} result in an increase in Lg.
3) After all the nodes in P_{init} are processed, we choose the next hop by choosing the neighbor that maximises Lg starting from the last problem that was solved.

The algorithm continues until the HammingWeight[Ct AND Tp]=HammingWeight[Tp].

Learning multiple topics: If multiple topics are to be learnt, then the target tuple is simply set to the OR of all the individual target tuples for every topic.

IV. Problem Networks in a Quantum Cryptography Game: QuaSim

We describe how the proposed approach can be used to design problem networks in our quantum cryptography game, *QuaSim*. QuaSim is a multiplayer, 3D serious game developed using the Unreal Engine including several problems involving quantum computing basics and cryptography protocols. QuaSim is an adaptive game that analyzes player interactions on-the-fly and adapts the game in several ways including providing several forms of hints, varying the difficulty of problems, teleporting players to the next interesting problem and so on. QuaSim has been played by over 100 students whose experiences are reported in [4]–[6].

Each play session in QuaSim is divided into a series of lessons which are further subdivided into problems. The first lesson in QuaSim introduces the concept of qubit (quantum analog for a classical bit) programming through polarization of photons. This lesson is divided into twelve problems designed by one of the authors who is a quantum cryptography expert. In each problem, the player has to activate one or more qubit receptors by programming a qubit using the correct orientation in a specified notation and basis. The number and the types of qubit receptors that the player activates depends on the topic being learned in the session. The associated set of concepts $C = \{S, Ae, Or, V, K, L, B\}$, where S= same angle, Ae= angle equivalency, Or= orthogonal, V= vector notation, K= ket notation, L= linear combination, and B= basis. Table I defines these concepts further.

The concept hierarchy CH is depicted in Figure 2. Note that the concepts S, Ae, Or, V don't have any dependencies.

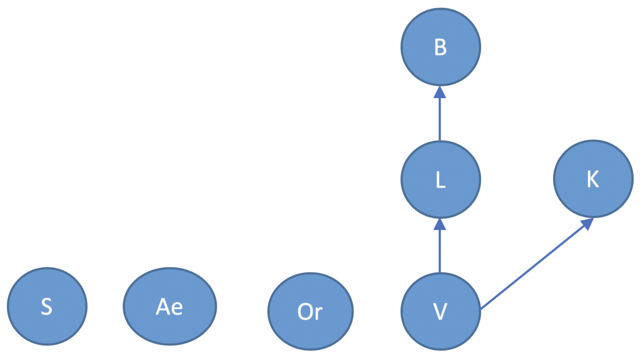


Fig. 2. The Concept Hierarchy Graph CH.

Assume that the concept tuple is given by Ct $=<S, Ae, Or, V, K, L, B>$ where the concept symbols are replaced with 1 or 0 depending on the condition discussed above. Initially, it is consists of all 0s. We will drop the commas between 1s and 0s for convenience.

In lesson 1 of QuaSim, each problem consists of two concepts $k = 2$, and there are 12 problems (expert input) with a replication factor of 3. The concept distribution on these problems is given by, $P_1 = \{S, V\}$, $P_2 = \{Ae, V\}$, $P_3 = \{Or, V\}$, $P_4 = \{S, K\}$, $P_5 = \{Ae, K\}$, $P_6 = \{Or, K\}$, $P_7 = \{S, L\}$, $P_8 = \{Ae, L\}$, $P_9 = \{Or, L\}$, $P_{10} = \{S, B\}$, $P_{11} = \{Ae, B\}$, $P_{12} = \{Or, B\}$.

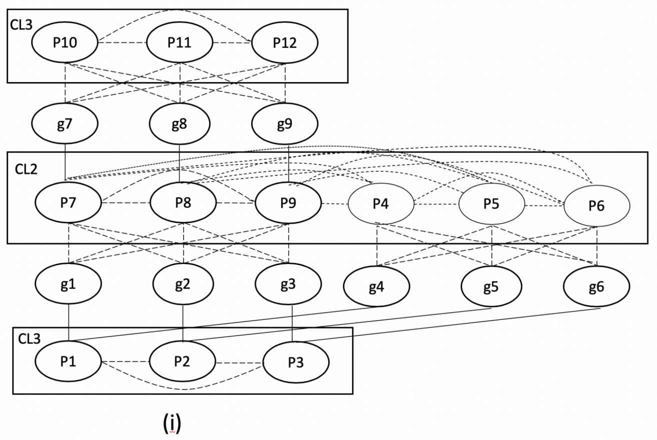
In other words, problem P_1 covers concepts same angle and vector notation, problem P_{10} covers concepts same angle and basis and so on.

Given the concept hierarchy CH, the problem dependency sets for the twelve problems are computed as: $D(P_1)$ = $D(P_2)$ =$D(P_3)$ = {}; $D(P_4)$ = $D(P_5)$ = $D(P_6)$ = $D(P_7)$ =$D(P_8)$ =$D(P_9)$ = {P_1, P_2, P_3}; $D(P_{10})$ = $D(P_{11})$ = $D(P_{12})$ = {P_7, P_8, P_9}.

The network construction algorithm takes the CH and the problem dependency sets as its inputs and outputs the problem network graph depicted in figure 3. The graph consists of twelve learning problem nodes corresponding to the twelve problems and 9 game nodes. Navigation edges are denoted by dashed lines and the game edges are denoted by solid lines in the figure. The nodes P_1-P_3 are interconnected by navigation

TABLE I
CONCEPT DEFINITIONS.

Concept	Definition/Student activity
S	Student creates a qubit by polarizing a photon at a particular angle (or the same angle that is shown to her).
Ae	Student learns the concept of angle equivalency, i.e. two photons polarized at an angle 180° from each other are equivalent.
Or	Student is introduced to the concept of orthogonal qubits, i.e. for example photons that have orthogonal polarization.
V	Student learns the vector notation for a qubit.
K	Student learns the ket notation for a qubit.
L	Student learns how to represent a qubit as a linear combination of linearly independent vectors.
B	Student learns the concept of basis.



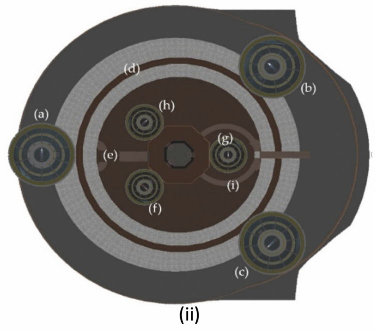


Fig. 3. Problem Network in Quantum Game: i) Problem network graph and problem cliques, ii)Spatial map of problem design in QuaSim derived from Network mapping –(a)P1 (b)P2 (c)P3 (d)Walkway (e)Stairs (f)P4 (g)P5 (h)P6 (i)Teleportation portal to third level.

edges as there dependency sets are identical. Similarly, the nodes P_4-P_9 are interconnected and the nodes P_{10}-P_{12} are interconnected due to their identical dependency sets. The game nodes g_1-g_3 capture the dependencies between problems P_1-P_3 and problems P_4-P_7 respectively, induced by the dependency between the concepts V and K. The game nodes g_4-g_6 capture the dependency between concepts V and L and the nodes g_7-g_9 capture the dependency between concepts L and B.

The network mapping algorithm takes the problem network graph and the problem dependency sets as its inputs and outputs three cliques – CL_1-CL_3 depicted in Figure 3(i). The problem node partition output by the algorithm is: $\{P_1,P_2,P_3\}$, $\{P_4,P_5,P_6,\ P_7,P_8,P_9\}$ and $\{P_{10},P_{11},P_{12}\}$. The enabling

TABLE II
NODE TUPLES FOR PROBLEMS IN EXAMPLE 2. IF A PROBLEM COVERS A PARTICULAR CONCEPT, THE TUPLE HAS A 1 IN THE CORRESPONDING POSITION IN THE TUPLE.

P_1	$< 1001000 >$
P_2	$< 0101000 >$
P_3	$< 0011000 >$
P_4	$< 1000100 >$
P_5	$< 0100100 >$
P_6	$< 0010100 >$
P_7	$< 1000010 >$
P_8	$< 0100010 >$
P_9	$< 0010010 >$
P_{10}	$< 1000001 >$
P_{11}	$< 0100001 >$
P_{12}	$< 0010001 >$

sets output by the algorithm are: $E(g_1)$=$E(g_4)$=$\{P_1\}$; $E(g_2)$=$E(g_5)$= $\{P_2\}$; $E(g_3)$=$E(g_6)$=$\{P_3\}$; $E(g_7)$=$\{P_7\}$; $E(g_8)$=$\{P_8\}E(g_9)$=$\{P_9\}$.

These are used by the QuaSim game designers to distribute the problems P_1-P_3 across the first level of a building in the game. Problems P_5-P_9 can be placed in a single level as well. As a design layout optimization, the designers distributed problems P_4-P_6 distributed at the second level of the game and placed problems P_7-P_9 in the third level of the game. The final three problems P_{10}-$P12$ were placed together at the top level of the building.

The three game nodes between the groups CL_1 and CL_2 were used to design physical barriers between first level and the second and third levels. There are no physical barriers between the second and the third levels of the building allowing players to freely move between the problems at these levels. The game nodes between CL_2 and CL_3 were used to design physical barriers between the second and third levels and the top level. As a design layout optimization, the three physical design barriers specified by the mapping algorithm were optimized into two game elements – a stair and a walkway. The stairs are enabled whenever either of the two problems P_1 or P_2 is solved. The walkway is enabled whenever the problem P_3 is solved. The physical barrier layout were similarly optimized for the other levels. For instance, there is a physical barrier between the third level and the top level that is enabled only upon solving any of the problems in the third level. The QuaSim design layout for the first two levels is depicted in figure 3(ii).

A. Network Traversal

Assume that the player wants to learn a topic, namely 'qubit representation with respect to a basis' and the concepts that need to be covered for this include S and B. However, B depends on L and L depends on V. Therefore, a student needs to cover S, V, L, and B. As a result, the target tuple is given by $< 1001011 >$.

Although not required for our navigation algorithm a quick glance at the twelve problems tells us that in order to acquire these concepts the student needs to solve three problems: P_1, and one from P_7, P_8, P_9 and another from P_{10}, P_{11}, P_{12}. Therefore, a path of length three exists that covers all the concepts. In the following, we discuss how the network traversal filters out different problems such that the student is able to achieve proficiency in the said topic by solving the fewest number of problems.

The navigation algorithm first identifies the initial set of problems, P_{init}, which contains problems P_1, P_2, and P_3. Table II gives all the node tuples for the problems P_1 through P_{12}. Therefore, if a student solves problem P_1, Ct is updated to $< 1001000 >$. If the student solves P_2, Ct is updated to $< 0101000 >$ and so on. The network traversal algorithm guides the student through the problem network by maximizing the learning gain metric, Lg.

Table III illustrates the computation of learning gain for all the problems in the P_{init} set. Initial value of Ct is all 0s. As we can see that solving P_1 gives a learning gain of 2 while solving P_2 and P_3 gives a learning gain of only 1. The navigation algorithm therefore chooses P_1 as the first problem to be solved. This results in an updated concept tuple Ct of $< 1001000 >$.

Next the algorithm looks at the neighbors of P_1 which in this case are all the problems in the network except for problems P_{10}, P_{11} and P_{12}. Of the neighbors, first it checks if there are any in the P_{init} set that result in an increase in the learning gain. In this case problems P_2 and P_3 are in P_{init}. We can quickly see that these two problems do not provide any increase in the learning gain and hence are discarded. Next it checks the learning gains from problems P_4 through P_9. The learning gain calculations for these problems is given in table IV. Since, the learning gain was already at 2 and problems P_4, P_5 and P_6 do not provide any improvement in that value. However, problems P_7, P_8 and P_9 provide an equal increase. Therefore, we can choose any of these problems as the next hop. Let's assume that the player solves P_8, then Ct is updated to $< 1101010 >$.

Now, P_8 is connected to problems P_{10}, P_{11} and P_{12}. We see that each of these provide an equivalent learning gain since we acquire the concept B that we need. The resulting new concept tuple is then $< 1101011 >$.

At this point we see that the HammingWeight[Ct AND Tp]= HammingWeight[$< 1101011 >$AND$< 1001011 >$]=4 HammingWeight[Tp]. Therefore, the algorithm concludes and we have acquired all the concepts that we were interested in.

The serious game QuaSim has five lessons currently each involving multiple problems. The more involved lessons involving multiple players are divided into tens of exercises with each exercise having tens of lessons. The problems designed and implemented using the proposed method was played by over 100 players. Their learning and gaming experiences are reported in our earlier works (see [4]). Our focus in this paper is on the algorithms used to incorporate problems into games such as QuaSim.

V. Conclusions

In this paper, we have described a novel approach to designing a problem network and its mapping onto gaming elements in a serious game. The approach has been implemented in QuaSim, a quantum cryptography virtual educator built in Unreal Engine 4. The proposed approach is inspired by the design of P2P networks where nodes are mapped to problems, and resources on each node are mapped to concepts that a problem covers. The network traversal (resource location) in P2P networks in order to say collecting different parts of a movie file is akin to collecting concepts that help learn a particular topic. QuaSim was played by over 100 students at our University and results of our observations have been reported earlier [4]–[6]. This paper focused on the design and incorporation of problems into serious games and described a novel approach for systematic design and incorporation of problem networks into game scenarios so that players can learn a topic of interest by efficiently navigating to and solving relevant problems. In future work, we plan to combine the notions of probabilistic knowledge structure and stochastic assessment procedure [29] with the problem network models presented in this paper to model the partial learning of a concept and partial readiness to learn a concept. We also plan to investigate methods to incorporate new problems into and remove old problems from the network, on the fly, depending on student performance as they play the game.

References

[1] A. Labuschagne, N. Veerasamy, and I. Burke, "Design of cyber security awareness game utilizing a social media framework," *2011 Information Security for South Africa*, pp. 1–9, 2011.

[2] B. D. Cone, M. F. Thompson, C. E. Irvine, and T. D. Nguyen, "Cyber security training and awareness through game play," in *Security and Privacy in Dynamic Environments*, S. Fischer-Hübner, K. Rannenberg, L. Yngström, and S. Lindskog, Eds. Boston, MA: Springer US, 2006, pp. 431–436.

[3] K. Boopathi, S. Sreejith, and A. Bithin, "Learning cyber security through gamification," *Indian Journal of Science and Technology*, vol. 8, no. 7, 2015.

[4] D. Abeyrathna, S. Vadla, V. Bommanapally, M. Subramaniam, P. Chundi, e. G. A. Parakh", M. Allegra, and H. Söbke, "Analyzing and predicting player performance in a quantum cryptography serious game," in *Games and Learning Alliance*. Springer International Publishing, 2019, pp. 267–276.

[5] A. Parakh, M. Subramaniam, and E. Ostler, "Quasim: A virtual quantum cryptography educator," in *2017 IEEE International Conference on Electro Information Technology (EIT)*, May 2017, pp. 600–605.

[6] S. Vadla, A. Parakh, P. Chundi, and M. Subramaniam, "Quasim: A multi-dimensional quantum cryptography game for cybersecurity," in *22nd Colloquium for Information Systems Security Education (CISSE)*, June 2018.

[7] R. D. Hoehn, N. Mack, and S. Kais, "Using quantum games to teach quantum mechanics, part 2," *Journal of Chemical Education*, vol. 91, no. 3, pp. 423–427, 2014.

TABLE III
COMPUTATION OF LEARNING GAIN IN THE INITIAL PROBLEM SET P_{init}.

I= (Node tuples for Problem P_i) OR Ct	I	l = I AND Tp	Learning Gain Lg = Hamming Weight(l)
Nt_1 OR Ct	$< 1001000 >$	$< 1001000 >$	2
Nt_2 OR Ct	$< 0101000 >$	$< 0001000 >$	1
Nt_3 OR Ct	$< 0011000 >$	$< 0001000 >$	1

TABLE IV
COMPUTATION OF LEARNING GAIN FOR VIABLE NEIGHBORS OF P_1. NOTE THAT THE CURRENT VALUE OF CONCEPT TUPLE IS $< 1001000 >$.

I= (Node tuples for Problem P_i) OR Ct	I	l = I AND Tp	Learning Gain Lg = Hamming Weight(l)
Nt_4 OR Ct	$< 1001100 >$	$< 1001000 >$	2
Nt_5 OR Ct	$< 1101100 >$	$< 1001000 >$	2
Nt_6 OR Ct	$< 1011100 >$	$< 1001000 >$	2
Nt_7 OR Ct	$< 1001010 >$	$< 1001010 >$	3
Nt_8 OR Ct	$< 1101010 >$	$< 1001010 >$	3
Nt_9 OR Ct	$< 1011010 >$	$< 1001010 >$	3

[8] Google and Caltech. Qcraft: Quantum games. Institute for Quantum Information and Matter. [Online]. Available: http://iqim.caltech.edu/outreach/quantum-games

[9] P. Migda, P. Hes, and M. Krupiski. Quantum game with photons. [Online]. Available: http://quantumgame.io/

[10] S. Ornes, "Science and culture: Quantum games aim to demystify heady science," *Proceedings of the National Academy of Sciences*, vol. 115, no. 8, pp. 1667–1669, 2018. [Online]. Available: https://www.pnas.org/content/115/8/1667

[11] A. Anupam, R. Gupta, A. Naeemi, and N. JafariNaimi, "Particle in a box: An experiential environment for learning introductory quantum mechanics," *IEEE Transactions on Education*, vol. 61, no. 1, pp. 29–37, Feb 2018.

[12] C. Tahan. Meqanic: The amazing quantum computer puzzle game. Laboratory for Physical Sciences (LPS), University of Maryland-College Park. [Online]. Available: http://www.meqanic.com/app/

[13] S. Akl. Quantum chess. School of Computing, Queen's University. [Online]. Available: http://research.cs.queensu.ca/Parallel/QuantumChess/QuantumChess.html

[14] Decodoku. NCCR QSIT, University of Basel. [Online]. Available: http://www.decodoku.com/

[15] Quantum cats. The Institute for Quantum Computing (IQC), University of Waterloo. [Online]. Available: http://quantumcats.ca/

[16] S. DeVore and C. Singh, "Development of an interactive tutorial on quantum key distribution," in *Physics Education Research Conference 2014*, ser. PER Conference, Minneapolis, MN, July 30-31 2014, pp. 59–62.

[17] S. Arnab, T. Lim, M. B. Carvalho, F. Bellotti, S. de Freitas, S. Louchart, N. Suttie, R. Berta, and A. De Gloria, "Mapping learning and game mechanics for serious games analysis," *British Journal of Educational Technology*, vol. 46, no. 2, pp. 391–411, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.12113

[18] D. Dicheva, C. Dichev, G. Agre, and G. Angelova, "Gamification in education: A systematic mapping study," *Journal of Educational Technology & Society*, vol. 18, no. 3, pp. 75–88, 2015. [Online]. Available: http://www.jstor.org/stable/jeductechsoci.18.3.75

[19] J. Hamari, D. J. Shernoff, E. Rowe, B. Coller, J. Asbell-Clarke, and T. Edwards, "Challenging games help students learn," *Comput. Hum. Behav.*, vol. 54, no. C, pp. 170–179, Jan. 2016. [Online]. Available: http://dx.doi.org/10.1016/j.chb.2015.07.045

[20] S. Heintz and E. L.-C. Law, "Digital educational games: Methodologies for evaluating the impact of game type," *ACM Trans. Comput.-Hum. Interact.*, vol. 25, no. 2, pp. 8:1–8:47, April 2018. [Online]. Available: http://doi.acm.org/10.1145/3177881

[21] M. Gaydos, "Seriously considering design in educational games," *Educational Researcher*, vol. 44, no. 9, pp. 478–483, 2015. [Online]. Available: https://doi.org/10.3102/0013189X15621307

[22] A. Slimani, M. Sbert, I. Boada, E. Fatiha, and M. Bouhorma, "Improving serious game design through a descriptive classification: A comparation of methodologies," *Journal of Theoretical and Applied Information Technology*, vol. 92, pp. 130–143, 10 2016.

[23] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Modeling and User-Adapted Interaction*, vol. 4, no. 4, pp. 253–278, Dec 1994. [Online]. Available: https://doi.org/10.1007/BF01099821

[24] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon, "Individualized bayesian knowledge tracing models," in *Artificial Intelligence in Education*, H. C. Lane, K. Yacef, J. Mostow, and P. Pavlik, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 171–180.

[25] Y. Long and V. Aleven, "Educational game and intelligent tutoring system: A classroom study and comparative design analysis," *ACM Trans. Comput.-Hum. Interact.*, vol. 24, no. 3, pp. 20:1–20:27, Apr. 2017. [Online]. Available: http://doi.acm.org/10.1145/3057889

[26] K. Smith, J. Shull, Y. Shen, A. Dean, and J. Michaeli, "Overcoming challenges in educational stem game design and development," in *Proceedings of the 2017 Winter Simulation Conference*, ser. WSC '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 60:1–60:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=3242181.3242246

[27] A. Gauthier, M. Corrin, and J. Jenkinson, "Exploring the influence of game design on learning and voluntary use in an online vascular anatomy study aid," *Comput. Educ.*, vol. 87, no. C, pp. 24–34, Sep. 2015.

[28] H. Sung, P. Wu, G. Hwang, and D. Lin, "A learning analytics approach to investigating the impacts of educational gaming behavioral patterns on students' learning achievements," in *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, July 2017, pp. 564–568.

[29] J.-P. Doignon and J.-C. Falmagne, "Knowledge Spaces and Learning Spaces," *arXiv e-prints*, p. arXiv:1511.06757, Nov 2015.