Computers & Education 126 (2018) 296–310

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

How to learn and how to teach computational thinking: Suggestions based on a review of the literature☆

Ting-Chia Hsu*, Shao-Chen Chang**, Yu-Ting Hung

Department of Technology Application and Human Resource Development, National Taiwan Normal University, 162, Sec. 1, East Heping Rd, Taipei City, 10610, Taiwan, ROC

ARTICLE INFO

Keywords:
Applications in subject areas
Pedagogical issues
Programming and programming languages
Teaching/learning strategies

ABSTRACT

Computational Thinking (CT) is seen as an important competence that is required in order to adapt to the future. However, educators, especially K-12 teachers and researchers, have not clearly identified how to teach it. In this study, a meta-review of the studies published in academic journals from 2006 to 2017 was conducted to analyze application courses, adopted learning strategies, participants, teaching tools, programming languages, and course categories of CT education. From the review results, it was found that the promotion of CT in education has made great progress in the last decade. In addition to the increasing number of CT studies in different countries, the subjects, research issues, and teaching tools have also become more diverse in recent years. It was also found that CT has mainly been applied to the activities of program design and computer science, while some studies are related to other subjects. Meanwhile, most of the studies adopted Project-Based Learning, Problem-Based Learning, Cooperative Learning, and Game-based Learning in the CT activities. In other words, such activities as aesthetic experience, design-based learning, and storytelling have been relatively less frequently adopted. Most of the studies focused on programming skills training and mathematical computing, while some adopted a cross-domain teaching mode to enable students to manage and analyze materials of various domains by computing. In addition, since the cognitive ability of students of different ages varies, the CT ability cultivation methods and content criteria should vary accordingly. Moreover, most studies reported the learners' CT performance and perspectives, while their information society ability was seldom trained. Accordingly, the research trends and potential research issues of CT are proposed as a reference for researchers, instructors, and policy makers.

1. Introduction

Computational thinking (CT) was first presented by Papert (1990), and since then its definition, teaching, and evaluation have been discussed by various scholars (Grover & Pea, 2013). Wing (2006) emphasized that CT is one of the daily life skills that everyone needs, rather than just being a programming skill used only by computer scientists. Wing (2010) further defined operational thinking as the process of problem-solving, so that the message processing agent can be effectively executed and the problem solved.

☆ This study is supported in part by the Ministry of Science and Technology in the Republic of China under contract numbers MOST 105-2628-S-003-002-MY3. This work is also partly supported by “Aim for the Top University Project” of the National Taiwan Normal University and the Ministry of Education, Taiwan, R.O.C.

* Corresponding author.
** Corresponding author.
E-mail addresses: ckhsu@ntnu.edu.tw (T.-C. Hsu), ntnuresearcher@gmail.com (S.-C. Chang).

https://doi.org/10.1016/j.compedu.2018.07.004
Received 20 September 2017; Received in revised form 29 June 2018; Accepted 2 July 2018
Available online 03 July 2018
0360-1315/

Computers can help us solve problems via the following two steps: First, consider the steps to solve the problem, then use technical skills to control the computer to help solve the problem. For example, one must understand mathematical formulas and explain the problem, and use simple methods or formulas to solve the problem via the computer's computation. Besides, when creating animation, the designer must plan the story and ways of shooting before drawing the computer animation, and complete the task using the computer software and hardware. In these two examples, CT is the thinking process one needs to engage in before beginning the computer and machine operation.

CT describes the processes and methods used to operate a system. It focuses on how people solve or research problems using computers (Wing, 2008), rather than on the computer hardware or imitating the computer's thinking mode. Moreover, Wing (2008) considered that CT is not only the center of problem solving, but also develops and identifies the problems. It is also reminiscent of early innovation in STEM and other subjects (Cheung, 2013), meaning that it not only lets the computer understand the way to solve problems, but also helps people to understand the solutions and problems. In other words, CT does not necessarily require machines, but people can produce CT processes by manipulating machines (Wing, 2008). Therefore, Wing (2008) considered that CT is no longered just essential for the learners in the departments related to computer science, but also indispensable to the learners in other domains. The current educators have to create and promote facilities for learning computational thinking.

With regard to the development status of CT, of the 21 countries in Europe, 17 participated in a survey which indicated that many of them were attempting to incorporate CT courses into their K-12 education curricula (Balanskat & Engelhardt, 2014). For example, the UK has implemented a complete set of CT courses, including computer science, information technology, and digital literacy throughout all disciplines (Brown, Sentance, Crick, & Humphreys, 2014). However, not all countries use computing in all subjects. Several countries have introduced an alternative to CT. The main reasons are: 1. Many teachers have been familiar with the teaching process and methods for many years, so it is difficult for them to change the original curriculum and adopt the new teaching content. However, in order to improve students' learning interest, teachers need to adopt new teaching methods. 2. To improve students' practical experience, teachers should apply programming in different subjects to improve the lower-achieving students' interest and level of achievement. 3. CT courses have become increasingly important in various fields, and many teachers have begun to introduce CT learning concepts in other courses (Angeli et al., 2016).

Take Australia as an example. A CT course was introduced into Australia's primary and secondary school curricula several years ago, and CT training was set up as one of its national teaching courses (Falkner, Vivian, & Falkner, 2014). According to the original promotion of combining digital technology (DT) courses with CT courses, the DT course is a multidisciplinary course that includes such subjects as English, mathematics, science, and art. From the basic training of first grade to the program development courses of ninth grade, children become familiar with using technology to solve complex problems and abstract concepts over a number of years (Armoni, 2012). Another example is Poland, where computer courses were divided into three phases in 1999. The first stage begins by training elementary school students in basic computer writing, painting, and reading. In the second stage, secondary school students are trained in the ability of computer computing, CT, and problem-solving. By the third stage, the computer course is one of the important subjects for the high school final examinations (Sysło & Kwiatkowska, 2015). The main goal of these three stages is to help students understand and analyze problems, use computers or other computer equipment to solve problems, and also apply CT to society or to their own lives.

South Korea has also developed a new curriculum for students. They started to promote computer education courses in 1971, with more than 34 h of computer courses in each grade of K-12 (Heintz, Mannila, & Färnqvist, 2016). At first, they only focused on teaching computer theory and the concepts of information science, but later they changed the curriculum to include the training of children's digital literacy, CT, and programming skills. South Korea has always been a textbook-based education culture; thus, they also reorganized the textbook for students in 2018 (Heintz et al., 2016).

From the CT literature review, it can be seen that it is currently an important subject of national education in many countries. Some countries have even classified CT as a national program or have prepared new teaching content and textbooks. From childhood, children have been trained in the abilities of CT, independent thinking, and problem solving. CT is no longer just an independent discipline or an independent area of teaching, but can also be applied to different disciplines or to daily life.

As for the future study directions of CT, the benefits of the implementation of CT have been widely recognized by scholars and educators. However, it is more important to think about how to successfully promote CT learning activities (Denning, 2017). At the same time, Denning (2017) mentioned that teachers were already familiar with the original teaching methods, and it was very difficult for them to change their teaching materials in a short period of time. For example, mathematics teachers are accustomed to solving mathematical problems using formula. The child's learning situation is to copy the teacher's formula so as to be able to come up with the correct answer in the examination. However, this approach limits the students' thinking and logic in the learning process. Denning (2017) proposed two steps to help educators and scholars examine the implementation of the situation. The first step emphasizes the educator's CT cognition. It can activate CT teaching activities completely, because of the correct concept of CT. The next step is learning about the CT assessment methods. If the teachers can correctly assess the effectiveness of students' CT, it is helpful for them to design learning activities and modify their teaching strategies. Simultaneously, teachers can help the students to complete the learning tasks according to their learning progress.

Heintz et al. (2016) indicated that CT is conducted in computer courses or other courses to train students' CT in many different countries. It is difficult to model or copy the CT development methods because of the differences in the national educational systems and culture. Therefore, many countries have begun to cultivate CT capability, programming ability, and digital skills in elementary school. However, the key to implementing CT is the teachers who have to train the students. The government needs to train the teachers in how to design CT activities and learning content so that the students can actively participate in the activities, improve their high-level thinking, and apply CT capabilities to other subjects (Orvalho, 2017).

From the CT literature review, the implementation of CT is still continuing to develop in many countries. Nevertheless, different aspects of performance have not been clarified in the past studies, including the countries, the age of learners, teaching strategies, and learning effectiveness. Therefore, in order to understand the development and application of CT in education, including subjects, age groups, learning strategies, and programming languages, in this study, we analyzed the related CT literature from 2006 to 2017. Possible research trends and issues are then proposed as a reference for future research in this area.

2. Literature review

2.1. Definitions of CT

Computational thinking is a universal skill in our life; it is no longer just the stereotypical impression of the skill required by computer engineers. We should all have a positive attitude towards, understand, and use this skill in our everyday lives (Wing, 2006). The abilities and limitations of CT are based on the processing of computing, regardless of whether it is people's minds or computers which are used to process the problem. In the early learning stage, children should not only be trained in the ability of the 3Rs (reading, writing, and arithmetic), but should also be taught how to put CT into practice and how to perform logical analysis (Wing, 2006). There are four CT operational skills, namely simplifying, embedding, transforming, and simulation. To transform a problem into an easy-to-understand problem (Wing, 2006), CT uses the basic concepts of computer science to solve problems, design systems, and change them into a thinking mode that can be understood by humans (Wing, 2006). Simultaneously, CT allows us to adopt a thinking mode similar to that of a computer scientist when facing problems (Grover & Pea, 2013).

Wing (2008) further defined CT as: 1. a conceptualization rather than the development process of programming language. Therefore, the students are asked to apply multiple layers of abstract thinking. CT is not restricted to using computers to learn (Wing, 2008); 2. a logical process rather than just repetitive behavior of mechanical operations. Therefore, people can be more flexible in exerting their own expertise through CT; 3. a way of human thinking, not the computer's calculation mode. In other words, CT is the way to solve human problems, not just copying the computer's thinking mode, because humans are smarter and more imaginative than computers (Wing, 2008); 4. a combination of mathematical thinking and engineering thinking to extend the foundation of mathematics; 5. a finished product of thinking, which helps us solve the problems in our life, manage the behaviors of daily life and the skills of communication and interaction with others; and 6. a basic skill in daily life, not an abstract philosophy. Because many scholars have different CT definitions and application methods, the taxonomies of CT from the recent 10 years are presented in the following section.

2.2. Taxonomies of CT

According to Wing (2006), CT can be classified into 11 thinking processes, including abstraction, algorithm design, decomposition, pattern recognition, and data representation. We also added the other steps of computational thinking that we found in the past studies, as shown in Table 1.

In the past 10 years, CT has been applied in different subjects. Scholars have attempted different learning strategies to help students learn. As Table 2 shows, this study lists the learning strategies that have been used in the past studies, including problem-based learning, collaborative learning, project-based learning, game-based learning, scaffolding, storytelling, computational learning theory, aesthetic experience, concept-based learning, embodiment-based learning, human-computer interaction teaching, and universal design for learning (Table 3).

For example, in terms of aesthetic experience, Farris and Sengupta (2016) attempted to use the ViMAP software to train fifth-grade students' aesthetic experience in a physics course. Students operate the program settings to understand the relationship between distance, speed, and acceleration. In addition, they combine the pitch and rhythm of the music with the speed and acceleration in physics. The software trained the students' physical concepts and aesthetic experience. In terms of game-based learning, Kazimoglu, Kiernan, Bacon, and MacKinnon (2012, p. 316) combined game and programming courses to help students learn the process of programming, problem solving skills, and CT logic. The research results showed that the students had good performance in learning outcomes and learning motivation. Moreover, this approach also promotes students' interaction and discussion with peers. In addition, Chang (2014) also attempted to use two visual programming software packages to explore 45 college students' learning anxiety and participation, and the system reliability and effectiveness in computer science courses. From the research results, it was found that the game-based learning approach can increase students' interest, and they were also satisfied with this learning activity. In terms of cooperative learning strategies, Wilkerson-Jerde (2014) designed a constructive cooperative learning environment for teaching students how to calculate area. The secondary students manipulated Java images through a collaborative system in the learning activity. Therefore, they could understand the change in area when they performed graphics conversion. From the experiment results, the students tried to dominate and assign roles in the process of cooperation, completing the learning task through cooperation. In addition, Pellas and Peroutseas (2017) combined Scratch and Second Life into mixed learning models to help students effectively learn the basic concepts of programming. According to Papert's framework of constructivist learning, students complete learning tasks through interaction and cooperation in virtual game environments. The research results showed that the framework of constructivist learning could help students learn social interaction, cognition, high-level thinking, and CT.

Table 1
The classification of computational thinking.

Num.	Thinking steps	Definition	Resource
1.	Abstraction	Identifying and extracting relevant information to define main ideas.	(Barr & Stephenson, 2011; Grover & Pea, 2013; Wing, 2006)
2.	Algorithm Design	Creating an ordered series of instructions for solving similar problems or for performing a task.	(Barr & Stephenson, 2011; Grover & Pea, 2013)
3.	Automation	Having computers or machines do repetitive tasks.	(Fletcher & Lu, 2009; Forrest & Mitchell, 2016; Kafai & Burke, 2013)
4.	Data Analysis	Making sense of data by finding patterns or developing insights.	(Angeli et al., 2016; Atmatzidou & Demetriadis, 2016; Basu, Biswas, & Kinnebrew, 2017; Cesar et al., 2017; Choi, Lee, & Lee, 2016; Magana & Silva Coutinho, 2017)
5.	Data Collection	Gathering information	(Barr & Stephens, 2011; CSTA, 2011)
6.	Data Representation	Depicting and organizing data in appropriate graphs, charts, words, or images.	(Benakli et al., 2017; Gynnild, 2014; Manson & Olsen, 2010; Stefan, Gutlerner, Born, & Springer, 2015; Weintrop et al., 2016)
7.	Decomposition	Breaking down data, processes, or problems into smaller, manageable parts.	(Kilpeläinen, 2010)
8.	Parallelization	Simultaneous processing of smaller tasks from a larger task to more efficiently reach a common goal.	(Barr & Stephenson, 2011)
9.	Pattern Generalization	Creating models, rules, principles, or theories of observed patterns to test predicted outcomes.	(ISTE & CSTA, 2011)
10.	Pattern Recognition	Observing patterns, trends, and regularities in data.	
11.	Simulation	Developing a model to imitate real-world processes.	(Barr & Stephenson, 2011; Grover & Pea, 2013; Wing, 2006)
12.	Transformation	Conversion of collection information.	(Wing, 2006)
13.	Conditional logic	Finding the associated pattern between different events.	(Grover & Pea, 2013)
14.	Connection to other fields	Finding the relationships between information.	(CSTA, 2011)
15.	Visualization	Visual content is easier to understand	
16.	Debug & error detection	Find your own mistakes and fix them	(Atmatzidou & Demetriadis, 2016; Berland & Lee, 2012; Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014)
17.	Efficiency & performance	Analyze the efficiency of the final results in order to achieve a more perfect goal.	(Grover & Pea, 2013)
18.	Modeling	Solve the current problems through the model architecture or develop a new system.	(Barr & Stephenson, 2011; ISTE & CSTA, 2011)
19.	Problem solving	The final step of logical thinking.	(Kim & Kim, 2016; Ngan & Law, 2015)

2.3. Subjects of CT

CT can be combined with various subjects, but many teachers are still using programming languages to teach it (Lye & Koh, 2014; Zhong, Wang, Chen, & Li, 2016). Many educators have argued that programming languages are the easiest and most appropriate ways to teach CT. However, such stereotypical ideas could confine the potential of logical thinking to a small percentage of subjects and learners (Wing, 2006, 2008). In fact, CT has been widely used in different subjects, including mathematics (Benakli, Kostadinov, Satyanarayana, & Singh, 2017; Snodgrass, Israel, & Reese, 2016; de Freitas, 2016), biology (Dodig-Crnkovic, 2011; Libeskind-Hadas & Bush, 2013; Navlakha & Bar-Joseph, 2011; Rubinstein & Chor, 2014), computer science (Repenning, 2012; Shell & Soh, 2013; Repenning, Webb, Koh, Nickerson, Miller, Brand, … & Repenning, 2015; Grover, Pea, & Cooper, 2015), language (Evia, Sharp, & Pérez-Quiñones, 2015) and programming (Bers, Flannery, Kazakoff, & Sullivan, 2014; Kazimoglu et al., 2012, p. 316; Wolz, Stone, Pearson, Pulimood, & Switzer, 2011). For example, Libeskind-Hadas and Bush (2013) presented a BioComp model to guide university students in solving biology problems through CT and programming logic in a biology course. According to the research results, students' learning efficiency and learning interest were significantly improved. Furthermore, their study also proposed changing the content and materials of BioComp to other subjects in the future. Grover et al. (2015) combined a MOOC platform and the Scratch software to train students' ability of problem solving and CT in a computer course. There were 54 middle school students who participated in the activities of this 7-week course to train their cognition of computing structure and computer learning. The results showed that the students not only improved their CT learning performance, but also improved their logical thinking ability and mathematical skills. In the meantime, Snodgrass et al. (2016) used Scratch to help disabled students learn mathematics in a primary school in the United States. They combined Scratch and mathematics to teach students how to calculate the time. The results found that the disabled students could train their CT ability through Scratch. Bers et al. (2014) attempted to help kindergarten students by training their problem-solving ability and CT through programming with robots. From the experimental results, the kindergarten children were interested in CT, programming, and robot manipulation. These results changed the kindergarten teachers' thinking about the age limit for learning programming, and they also rethought the course direction and design.

From the CT literature review, it can be seen that the development of operational thinking is not only applied in computer programming, but can also be used in mathematics and biology to train students' logical concepts, CT, problem solving skills, and

Table 2
Categories of the 16 learning strategies in the CT learning activities adopted in this study.

Strategy	Explanation
1. problem-based learning	The definition of problem-based learning is helping students to set their own learning goals through a problem scene. Students will explore the learning solution by themselves, and report their own learning conclusions and feedback to the team. Problem-based learning is not only used to solve problems, but also to enhance students' understanding of new knowledge through appropriate questions (Wood, 2003).
2. collaborative learning (teamwork)	Group learning is divided into: collaborative learning and cooperative learning. In cooperative learning, partners split the work, solve subtasks individually, and then assemble the partial results into the final output. In collaborative learning, group members are required to complete the task together, negotiate, and share meanings relevant to the problem-solving task (Dillenbourg, 1999; Roschelle & Teasley, 1995).
3. project-based learning	Project-based learning (PBL) is a model that organizes learning around projects. Projects are complex tasks, based on challenging questions or problems, that involve students in design, problem-solving, decision making, or investigative activities; PBL gives students the opportunity to work relatively autonomously over extended periods of time, and culminates in realistic products or presentations (Jones, Rasmussen, & Moffitt, 1997).
4. game-based learning	Game Based Learning (GBL) is similar to Problem Based Learning (PBL), wherein specific problem scenarios are placed within a play framework (Barrows & Tamblyn, 1980). GBL can provide a Student-Centered e-Learning (SCeL) approach (Motschnig-Pitrik & Holzinger, 2002). Moreover, games include many characteristics of problem solving, e.g. an unknown outcome, multiple paths to a goal, construction of a problem context, collaboration in the case of multiple players, and they add the elements of competition and chance.
5. scaffolding	Scaffolding provides the framework of learning to help the students learn the new knowledge at the beginning. The purpose of scaffolding is to train the students to solve problems independently.
6. problem solving system	To find the solution to problems through logical or special methods, and to understand the goals of the problem and apply the appropriate abilities and methods to solve the problem.
7. storytelling	Pesola (1991, p. 340) suggested that storytelling is "one of the most powerful tools for surrounding the young learner with language." According to Isbell (2002), many stories that work well with children include repetitive phrases, unique words, and enticing descriptions. These characteristics encourage students to join in actively to repeat, chant, sing, or even retell the story. Much of the language children learn reflects the language and behavior of the adult models they interact with and listen to (Strickland & Morrow, 1989). "Listening to stories draws attention to the sounds of language and helps children develop a sensitivity to the way language works" (Isbell, 2002, p. 27).
8. systematic computational strategies	Systematic computational learning theory provides a formal framework in which to precisely formulate and address questions regarding the performance of different learning algorithms so that careful comparisons of both the predictive power and the computational efficiency of alternative learning algorithms can be made.
9. aesthetic experience	Aesthetic experience provides the means through which meanings that are ineffable, but full of feeling, can be expressed and understood, helping us to tolerate ambiguity, to discern subtle relationships, and to focus on details (Kokkos, 2010).
10. concept-based learning	Concepts are a way to organize and make sense of learning. The students try to define the attributive differences among different concepts. Other researchers have made use of concept-based models or graphic organizers. The model described here relies heavily on including attributes that can be generalized to multiple instances. The other concept depends on the definition of the concept of exclusion featuring a collection of example facts (Boudah, Lenz, Bulgren, Schumaker, & Deshler, 2000; Erickson, 1998; Kameenui & Carnine, 1998).
11. HCI teaching	Human-Computer Interaction teaching (HCI teaching) is suitable for all grades of college students to learn natural science, and is also a common online teaching method (McCoy & Ketterlin-Geller, 2004).
12. design-based learning	Design-based learning is integrated design thinking and processes in the curriculum, which can be applied to many subjects. It asks students to set up their own goals and to create ideas to achieve them.
13. embodied learning	Theories of embodied cognition argue that mental modal simulations in the brain, body, environment and situated actions are composed of central representations in cognition. Based on embodied cognition, body movements of performing natural science experiments can provide learners with external perceptions for better knowledge construction.
14. teacher-centered lecture	Students put all the focus on the teacher, and concentrate on lectures without collaborative learning activities. Students will not miss the key points through the teacher guiding all of the activities.
15. Critical computational literacy	A concept of "computational literacy" helps us better understand the social, technical, and cultural dynamics of programming. Critical computational literacy emphasizes how to use the computational method, and what can be done.
16. Universal Design for Learning	The basis of Universal Design for Learning (UDL) is grounded in emerging insights about brain development, learning, and digital media (Hitchcock, Meyer, Rose, & Jackson, 2002). It arouses the learners' interest through multiple methods of communication and expression.

deductive ability. In addition, we also found that not only middle or high school students can participate in CT learning activities; kindergarten children can also cultivate CT ability. Therefore, educators and scholars have indicated that CT is a very important topic for the future.

3. Method

3.1. Definition of resources

The database used in this study is SCOPUS. First, we used the keyword of computational thinking to search for paper topics, abstracts, and keywords in the database. This gave us a total of 1133 articles. Second, we set the search period from January 1, 2006

Table 3
The classification of teaching tools in the CT courses.

Teaching tools	Explanation
1. LOGO	LOGO is a computer programming language that is easy to learn and use. Students can use it to draw patterns, calculate and emit sounds, and it is also a new way for elementary students to learn the computer programming language.
2. LEGO	Lego is command box programming that combines building with the familiar LEGO bricks, using easy-to-use coding software, making coding fun and relevant for elementary and middle school students.
3. ViMAP	ViMAP programming language is an open-source programming language and modeling environment designed for the K12 science classroom. ViMAP also allows children to create their own programming commands.
4. MATLAB	MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment. It allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages (C, C++, C#, Java, and so on.)
5. Alice	Alice is an open-source object-based educational programming language with an integrated development environment (IDE). It uses the functions of drag and drop to create computer animations with 3D models.
6. Turtle Art	Turtle Art is software with a Logo-inspired graphic “turtle” that combines with Scratch-like snap-together visual programming elements and colorful art.
7. Scratch	Scratch is an online visual programming language developed by MIT Media Lab. Users can create online projects and make them into anything by coding with simple blocks.
8. Scratch4SL	S4SL is based on Scratch and is a new easy way to add behaviors and interactivity to your objects in Second Life. It uses the graphical programming language to create a project by dragging graphical blocks.
9. Code.org	Code.org is a website that includes free coding lessons and which also attempts to encourage teachers to include more computer science classes in the curriculum. The users use Blockly to write code. This is an interesting virtual computer programming language, like a markup language.
10. AgentCubes	AgentCubes is an educational programming language for kids to create 3D and 2D online games and simulations. It is a computational thinking tool to teach kids computational thinking through game and simulation design based on the Scalable Game Design curriculum.
11. Scalable Game Design	The Scalable Game Design is a curriculum to learn about computational concepts at the level of computational thinking that is relevant to game design as well as to computational science.
12. Java	Java is an open source computer-programming language. The main belief of Java is that it can run on all platforms that support Java without the need for recompilation.
13. C	C is an imperative computer programming language and a typical machine instruction, which provides a bridge to embed and operate systems with various types of application software.
14. C++	C++ is a compiled language, with implementations of it available on many platforms. The efficiency and flexibility of C++ has also been found useful in many other contexts.

to December 31, 2017. After setting the published time period, the search result showed that there were 1112 CT articles between these dates. We then set the type of article to published academic journal papers, academic journal papers (in press), and books, which gave a total of 262 journal papers or books. Finally, we excluded non-SCI and non-SSCI journal articles, giving 120 articles for the follow-up analysis. Two experienced researchers then read and categorized the papers based on the coding scheme. During the coding process, if there were inconsistent coding values, the researchers were asked to discuss them until an agreement was reached.

3.2. Data distribution

Fig. 1 shows the publication situation of CT papers from January 2006 to 2017. The earliest paper was written by Wing (2006), and explained the definition of computational thinking to help readers re-interpret CT. The number of CT articles from 2006 to 2017 grew steadily, with the original single article growing to 21 in 2016 and 44 in 2017. In Fig. 1, it can be seen that 120 papers were published from 2006 to 2017. Such a finding is reasonable since CT is a new index for educators to design learning activities. In the beginning, scholars defined CT and attempted to promote it, until it gradually evolved into implementation in the classroom. Moreover, they shared questions of importing CT into courses, and provided solutions for future study to design courses and activities. Meanwhile, the number of CT papers doubled from 2016. It has thus gradually gained the attention of scholars and educators,

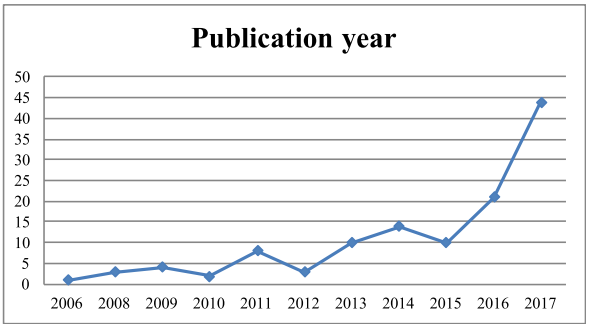


Fig. 1. The annual distribution of computational thinking papers.

and is a new topic that cannot be ignored in the future.

3.3. Coding scheme

The coding scheme was modified from the scheme developed by Hwang and Wu (2014), and included nationality, author, journal, application course, learning strategy, and participants (Hwang & Tsai, 2011; Hwang & Wu, 2014). In addition, we also collected relevant teaching content data, including teaching tools, programming language, and course categories (formal or informal). The following items explain the coding scheme for each dimension:

(1) Nationalities, authors, and journals:

The basic information of the published papers is discussed, including authors, nationality, and journals. The purpose is to understand which countries have more frequently published CT studies. Books and literature reviews were also included in the data collection project.

(2) Application domains:

The CT field includes programming, information engineering, computer application software, mathematics, biology, medicine, society, business management, language, music, computer science, journalism, robotics, science and technology, epidemiology, physics, STEM, social ecosystems, and algorithms.

(3) Learning strategy:

The category of 16 learning strategies was composed based on the previous studies in each country, and included project-based learning, problem-based learning, teacher-centered lectures, collaborative learning, game-based learning, aesthetic experience, concept-based learning, systematic computational strategies, scaffolding, problem-solving systems, storytelling, embodied learning, universal design for learning, HCI teaching, design-based learning, and critical computational literacy.

(4) Participants, programming languages and course categories (formal or informal):

This study reviewed the participants, teaching tools, programming languages, and course categories adopted in CT, including the age distribution of the participants, the category of programming language, and formal or informal courses.

(5) Teaching tools that are often used in CT courses:

The teaching tools that are often used in CT courses include programming software, games, mobile games, board games, experiments, Arduino, robots, Game Maker, video, IRS (Clickers) and e-books; however, in order to coordinate with the course, and considering suitability for different ages, the programming language categories include LoGo, LEGO, ViMap (based on Logo), MATLAB, ALICE, TurtleArt (similar to Scratch), Scratch, Scratch4SL (Scratch for Second Life), Code.org (similar to Scratch), AgentCubes (making 2D/3Dgames), Scalable Game Design, Java, C, and C++.

4. Research results

4.1. Nationalities, authors, and journals

In this research, we only listed the nationality information of the first author in the CT paper. Moreover, the meta-review method adopted was based on the concepts of systematic review as proposed by Wang, Liu, and Hwang (2017). According to the results of the current meta-review, many countries have begun developing CT instructional design. The distribution status of the top five countries can be seen in Fig. 2. Due to the selection pool of this study, the top four are the United States, Spain, Greece, and the United Kingdom.

In addition, this meta-review also analyzed the institutes that published more than two CT research papers including the United States (62), Spain (7), Greece (5), the United Kingdom (5), South Korea (4), China (4), Canada (3), Turkey (3), Taiwan (3) and Singapore (3).

4.2. Adopted strategies of CT

In the CT activity category, the distribution of learning strategies is shown in Fig. 3. Most of them are project-based learning strategies and problem-based learning strategies. A total of 22 papers used these two learning strategies to conduct CT activities. The next was the collaborative learning strategy, with a total of 16 papers. This was followed by game-based learning strategies, with 12 papers. However, 27 studies that were originally designed for case design or CT instructional design did not mention specific learning strategies in the study, and neither were they empirical studies. Therefore, they were not included in this category of data

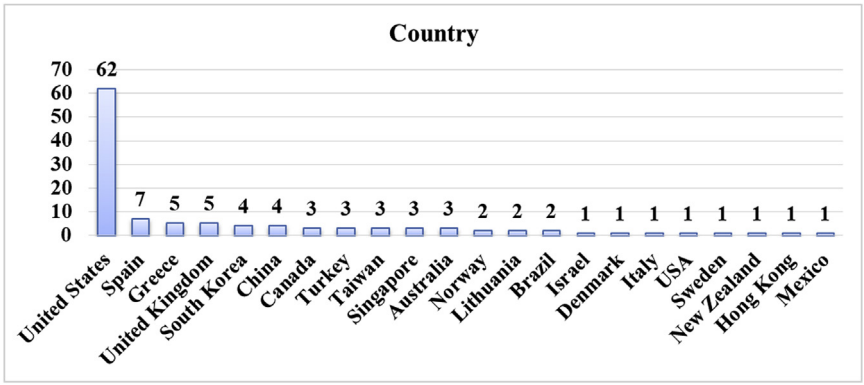


Fig. 2. The number of CT papers by country.

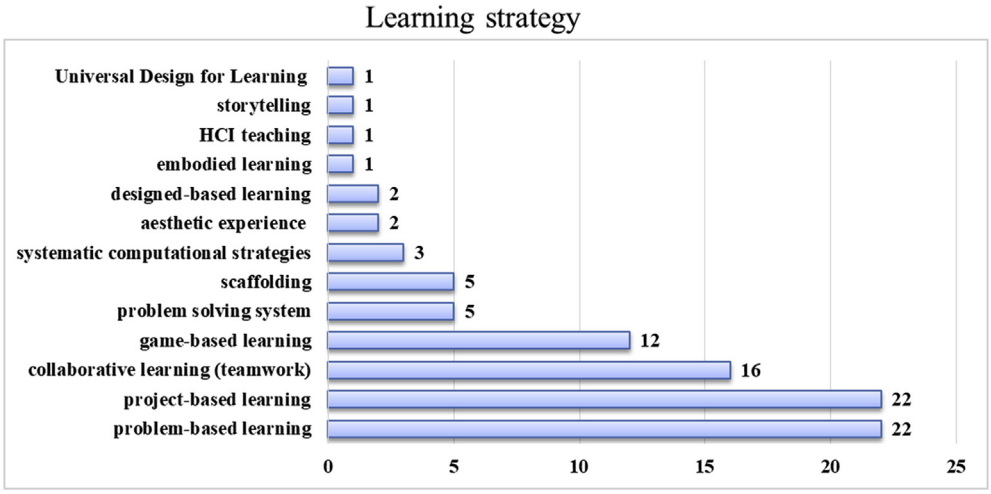
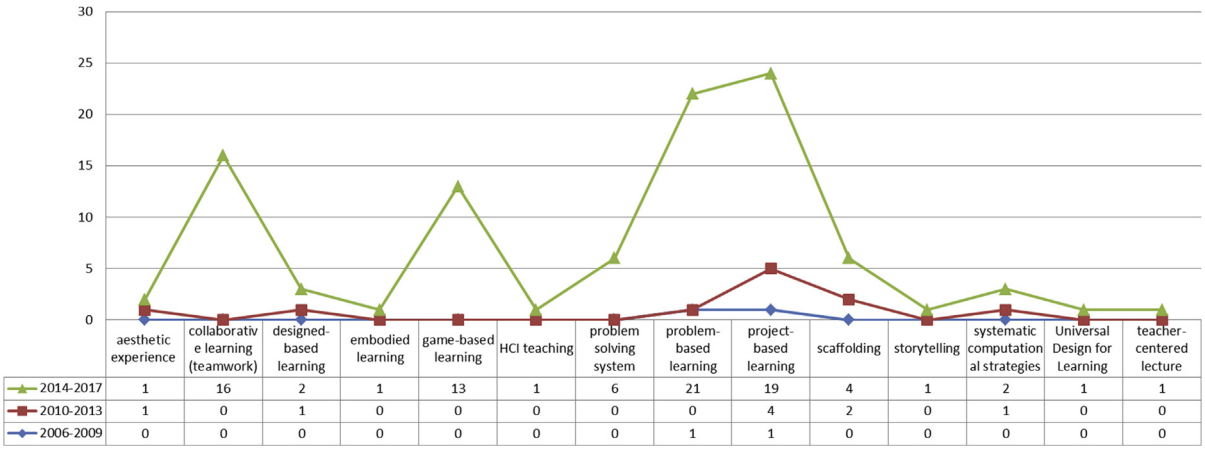


Fig. 3. The number of computational thinking studies for each learning strategy.

classification and statistical analysis. According to the results, CT activities are usually conducted as part of the topic. Students need to complete their learning tasks in the learning activities and try to solve problems with their own knowledge. They need to come up with the best answers through collaborative processes. Emphasis is placed on the process of knowledge application and learning



	aesthetic experience	collaborative learning (teamwork)	designed-based learning	embodied learning	game-based learning	HCI teaching	problem solving system	problem-based learning	project-based learning	scaffolding	storytelling	systematic computational strategies	Universal Design for Learning	teacher-centered lecture
2014-2017	1	16	2	1	13	1	6	21	19	4	1	2	1	1
2010-2013	1	0	1	0	0	0	0	0	4	2	0	1	0	0
2006-2009	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Fig. 4. The number of CT papers by learning strategy each year.

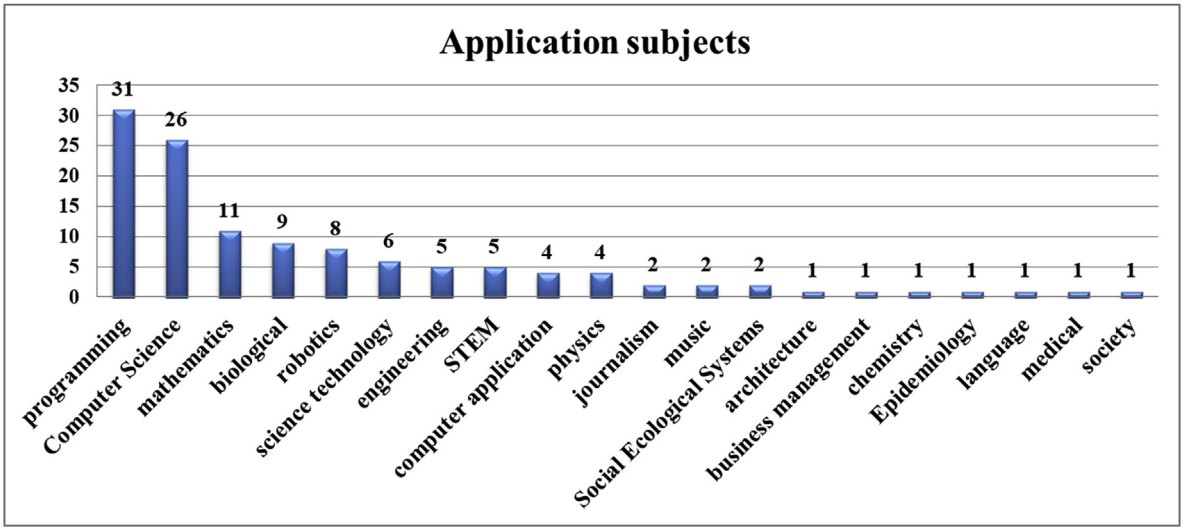


Fig. 5. The number of CT papers by application subjects.

interaction.

In this study, we divided the period into three stages, that is 2006–2009, 2010–2013, and 2014–2017, to discuss changes in learning strategy development. As indicated in Fig. 4, in the first stage, 2006–2009, two learning strategies were employed in CT activities, namely problem-based Learning and project-based Learning. More strategies were introduced into CT activities during the second stage, 2010–2013, when the major learning strategy was project-based learning with four CT papers, and the secondary strategy was scaffolding, with two papers. Such development may have been influenced by the popularization of computer technology that gave schools more chances to design and develop CT courses. In the third stage, 2014–2017, 14 learning strategies were included in the CT activities, which suggests that CT ability was gaining more attention and educators and researchers set about adopting different learning strategies to help students improve their learning performance through CT activities.

4.3. Application subjects

Next, we analyzed the subjects in which CT was applied. It can be discovered from Fig. 5 that the number of programming subject papers employing CT was 31, constituting the biggest proportion. Programming was followed by 26 computer science papers, 11 mathematics papers and 9 biology papers. There were also 28 papers about preliminary single-case design or suggestions for CT instructional design that neither showed a particular subject category nor counted as an empirical study and so were excluded from this section of data classification. Moreover, there are 30 papers using more than two subjects in a study. Therefore, it was found that CT was mostly applied to programming design and computer science courses, while a number of scholars also introduced it into different subjects such as biology, mathematics, language, and music. The results signify that CT is not only essential for computer-related subjects, but also for training the computing capabilities of mathematics, or cultivating problem-solving abilities in biology. Accordingly, different subjects could also employ the essential ideas of CT. As a result, how to apply CT to other subjects is the most important and challenging issue at present (Denning, 2017).

4.4. Teaching instruments, programming language and course type (formal and informal courses)

The employment of teaching instruments in CT activities is shown in Fig. 6. The most frequently used was the programming design course, included in 30 papers. The second was helping students complete CT activities by virtue of experiments and computer games. Meanwhile, robots, board games, IRS, videos, Game Maker or other instruments were also involved in events designed to train students' CT capability. However, a few papers did not adopt any teaching instrument and therefore could not be categorized for discussion in this section.

As suggested in Fig. 7, the most popular programming language which the teachers used for designing CT learning activities was Scratch. Scratch was applied in 19 research papers in total, followed by ALICE, ScratchASL, LEGO and others, since most teachers believed that visual programming design software was more likely to gain popularity among students (Chen et al., 2017; Grover & Pea, 2013). We also found that there were seven papers teaching two programming languages in their study. Another 81 papers did not apply any programming language in CT teaching and therefore were not included in this section's analysis. It can be concluded that currently there are numerous programming design tools to help develop students' CT capability. With instantaneous interactive programming design tools, students' logical thinking ability may be exercised, and the best solution may be found by multiple tries to operate the software.

As for the analytical results of course types shown in Fig. 8, the percentage of CT articles related to formal courses was 26, while

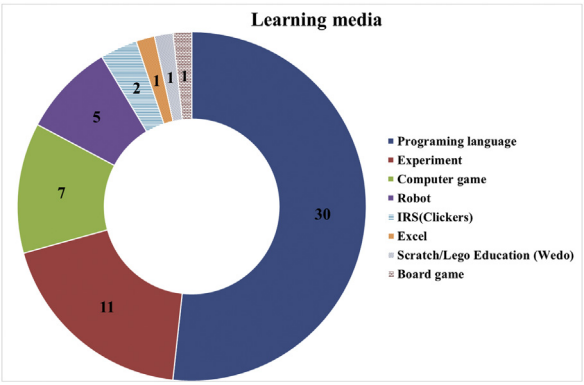


Fig. 6. The number of CT papers by teaching instruments employed.

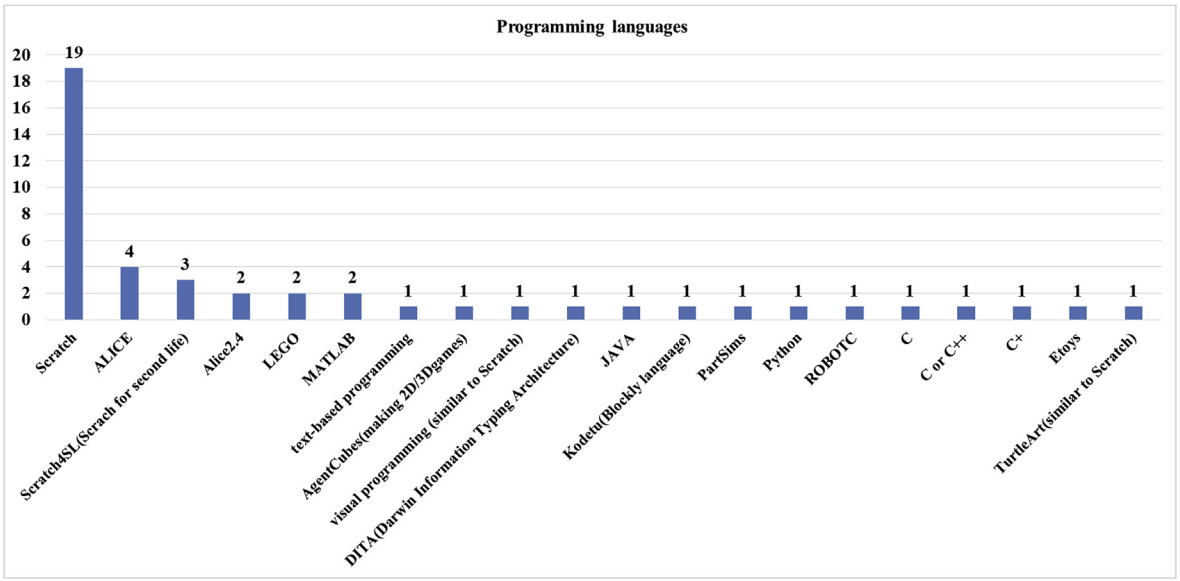


Fig. 7. The number of CT papers by programming languages adopted.

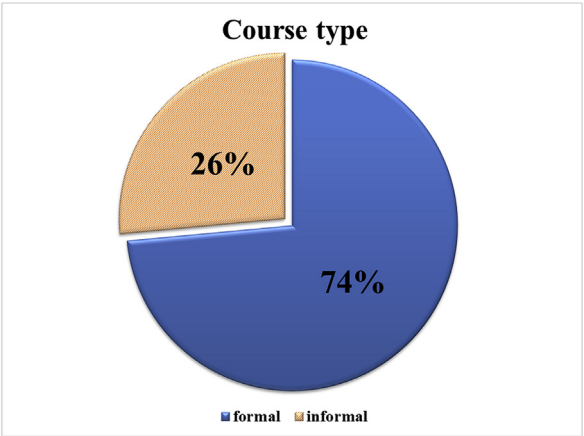


Fig. 8. The percentage of CT papers by course type.

Table 4
The number of CT papers by age group and subject.

Subject/Age group	preschoolers	elementary school	secondary	senior high	university	adults
biology	0	1	0	0	2	2
chemistry	0	0	0	1	0	0
computer application	0	0	1	1	2	0
computer science	0	4	3	5	9	0
engineering	0	1	0	0	0	2
language	0	0	0	0	1	0
mathematics	1	0	0	4	3	1
music	0	1	0	0	0	0
physics	0	0	1	3	1	0
programming	2	12	7	3	1	2
robotics	0	3	1	1	1	0
science technology	0	1	1	1	0	2
social ecological systems	0	0	1	0	0	0
STEM	0	2	2	0	0	0

there were 74% for informal courses, implying that most research aimed to include CT in formal courses. Furthermore, a number of countries list CT among the national curriculum subjects (Falkner et al., 2014; Heintz et al., 2016); yet a small proportion of research still applied CT in informal courses, suggesting that it is an indispensable learning course for children apart from the teaching content of formal courses, which resulted in attempts to conduct CT learning activities in informal courses. In addition, another 44 papers about preliminary single-case design or suggestions regarding CT instructional design but not about actual and practical applications for teaching were excluded from this section's data classification.

4.5. The relationship between age and subject

In order to conduct in-depth analysis of the changes in the application subjects of CT learning activities by various age groups, comparisons were made regarding age groups and subjects, as indicated in Table 4. It can be found from the analytical results that programming design is a subject presently participated in by students of all ages and by adults, among which the largest group was elementary school students, followed by secondary school students and senior high school students, representing that nowadays children begin to take program design courses from elementary school, and teachers are expected to train students' CT ability through program design courses. In addition, subjects requiring specialized knowledge, such as biology and science & technology, were more often taken by students than by adults. Meanwhile, it was found that the most common subject among senior high students was computer science. Logical thinking, systematic thinking, and other CT could be cultivated by learning computer science-related knowledge and skills. The application ability of CT, such as problem solving ability, team cooperation, and innovative thinking ability, could be improved via the design and practice of information technology. Moreover, senior high school students should get further knowledge of the essence of information technology and internalize CT so as to develop the ability of innovative thinking and team cooperation.

4.6. The relationship between age and strategy

To further look into the relationship between ages and strategies in CT activities, the distribution of learning strategies throughout the age groups was collected and sorted as indicated by Table 5. As can be inferred from the table, problem-based learning and project-based learning were teaching strategies for which the coverage of age groups ranked the highest. Both of them were mainly

Table 5
The CT papers by age group and teaching strategy.

Teaching strategy/Age group	preschoolers	elementary school	secondary	senior high	university	adults
aesthetic experience	0	1	0	0	0	0
collaborative learning	0	5	4	1	4	0
design-based learning	0	2	0	0	0	0
embodied learning	0	0	0	1	0	0
game-based learning	0	4	4	3	1	0
problem solving system	0	1	2	1	1	0
problem-based learning	0	6	2	4	5	3
project-based learning	0	5	6	2	7	2
scaffolding	1	1	0	1	1	0
storytelling	0	1	0	0	0	0
systematic computational strategies	0	0	0	0	2	0
Universal Design for Learning	1	0	0	0	0	0
teacher-centered lectures	0	0	0	0	1	0

Table 6
The number of CT papers by teaching strategy and subject.

Teaching strategy/ Subject	architecture	biology	computer application	computer science	engineering	language	mathematics	music	physics	programming	robotics	science technology	social ecological systems	society	STEM
aesthetic experience	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0
collaborative learning	0	0	0	5	0	0	0	0	0	8	3	1	1	0	2
design-based learning	1	0	0	0	1	0	1	1	0	1	0	0	0	0	0
embodied learning	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
game-based learning	0	1	1	5	0	0	1	0	1	7	2	0	0	0	0
HCI teaching	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
problem solving system	0	1	0	2	0	0	3	0	2	1	0	0	0	1	0
problem-based learning	0	4	1	7	1	0	2	1	1	11	4	1	0	0	0
project-based learning	0	2	1	8	2	1	2	0	2	9	4	3	1	0	4
scaffolding	0	0	0	3	0	0	2	0	1	2	0	0	0	0	0
storytelling	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
systematic computational strategies	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0
Universal Design for Learning	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
teacher-centered lectures	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

applied to K-12 children, college students, and then adults, who may be motivated to participate in CT activities in this way. Moreover, it could guide children to seek answers in activities, exercise CT ability, and improve their learning performance. What is more, Game-based Learning, which may inspire students' imagination and creativity, and hence help improve CT ability, was adopted in the CT activities of junior high school and senior high school to stimulate students' interest in learning. On the other hand, Project-oriented Learning was often used with college students since the students' creativity and sense of responsibility could be invoked and team cooperation ability enhanced through the allocation of project tasks.

4.7. The relationship between subjects and strategies

Finally, we discuss the relationship between and distribution of subjects and strategies in CT activities. As shown in Table 6, problem-based learning was most commonly used in different subjects, with nine subjects employing it in CT activities, followed by project-based learning with nine subjects. On the other hand, the learning strategies employed by teachers to adapt to the instructional content of different subjects were entirely different. The most popular application of project-based learning and problem-based learning were program design courses, which may be the result of the need to assign homework or classroom tasks in the form of submitting projects that for the most part require students to complete a program project. Consequently, it can be inferred that many researchers begin to introduce various teaching strategies to each subject, aiming at enhancing students' CT ability and thus improving their learning performance with the help of teaching strategies.

5. Discussion and conclusions

In this study, CT papers published in academic journals between 2006 and 2017 were reviewed and analyzed and discussed by data classification. It was found that the number of CT papers has seen a substantial rise in recent years, and CT has received positive comments from scholars from a number of countries (Balanskat & Engelhardt, 2014; Falkner et al., 2014; Heintz et al., 2016; Sysło & Kwiatkowska, 2015), implying its considerable importance for achieving future educational goals.

It was revealed by the statistical analytical results that CT activities were mostly introduced to program design, computer science, biology, and robot design courses. It can be concluded that, so far, a number of CT activities have been integrated into different subjects in a life-based manner, echoing the CT concepts proposed by Wing (2006). He considered CT as a skill that could be widely applied in the living environment rather than being exclusively employed by computer engineers; in contrast, it is a skill that deserves a positive attitude in daily life and should be known about and engaged in. Therefore, CT is a topic worth in-depth study in the future, and the impact of CT on children's academic performance is also a worthy topic of discussion.

The instructional application of CT and learning strategies was also discussed; it was discovered that most research centered on Project-oriented Learning, Problem-oriented Learning, Cooperative Learning, and Game-based Learning. During the last decade, a number of research scholars have mentioned the benefits of CT for children's learning; hence, future research should attempt to introduce different learning strategies, including the Scaffolding Learning Strategy, Storytelling Learning, and Aesthetic Experience, so as to aid learners in multiple ways in terms of the development of subjects or high-level ability training, say, training in critical thinking and problem-solving ability.

As a summary of the above analytical results, suggestions for the design of future research on CT are as follows:

(1) Educate faculty about CT. In order to achieve a thorough introduction and design of CT courses, front-line faculty need to receive overall education and form correct concepts to carry out and invigorate their CT teaching.
(2) Effectively assess students' learning performance. Because different learning strategies and subjects will be applied at different ages, formal and informal courses also need different scoring guidelines. Such assessment may be of help later in designing teaching activities and modifying teaching strategies.
(3) Know about students' learning status. Teachers need to consider students' learning status when teaching so as so guide the students through the CT training courses, or offer proper aid or feedback regarding different students.
(4) Since the cognitive ability varies among students of different ages, the CT ability cultivation methods, content criteria, and learning strategies should vary accordingly. CT training courses should be designed for different ages using appropriate strategies.
(5) Adopt the cross-domain teaching mode to enable students to manage and analyze materials of various domains by computing, so as to deepen their comprehension of cross-domain knowledge, experience the roles played by cross-domain knowledge and computing in solving complicated problems in the real world, and foster their interest in science, technology, engineering, and mathematics.

This study aimed to review and analyze the development and changes in CT research in the last decade (from 2006 to 2017), and propose potential directions for future research. With the rapid growth in technological skills and computer information, the virtual world keeps merging with the real world, and digitization and computation of computers are evolving to be the basic features of modern society. To help students correctly understand and integrate into the information society, it is not enough to cultivate their creativity ability and improve their digital literacy; they also need to enhance their CT capability, learn to utilize new technological skills, and take full advantage of such skills to adjust to the rapid change in the information society. Therefore, how to design CT teaching and research, and how to combine proper learning strategies with subjects is an issue worth studying.

References

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., et al. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology & Society, 19*(3), 47–58.

Armoni, M. (2012). Teaching CS in kindergarten: How early can the pipeline begin? *ACM Inroads, 3*(4), 18–19.

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661–670.

Balanskat, A., & Engelhardt, K. (2014). *Computing our future: Computer programming and coding-priorities, school curricula and initiatives across Europe.* European Schoolnet.

Barrows, H. S., & Tamblyn, R. M. (1980). *Problem-based learning: An approach to medical education.* Springer Publishing Company.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *Acm Inroads, 2*(1), 48–54.

Basu, S., Biswas, G., & Kinnebrew, J. S. (2017). Learner modeling for adaptive scaffolding in a computational thinking-based science learning environment. *User Modeling and User-Adapted Interaction, 27*(1), 5–53.

Benakli, N., Kostadinov, B., Satyanarayana, A., & Singh, S. (2017). Introducing computational thinking through hands-on projects using R with applications to calculus, probability and data analysis. *International Journal of Mathematical Education in Science and Technology, 48*(3), 393–427.

Berland, M., & Lee, V. R. (2012). Collaborative strategic board games as a site for distributed computational thinking. *Developments in Current Game-Based Learning Design and Deployment, 285*.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education, 72*, 145–157.

Boudah, D. J., Lenz, B. K., Bulgren, J. A., Schumaker, J. B., & Deshler, D. D. (2000). Don't water down! Enhance content learning through the unit organizer routine. *Teaching Exceptional Children, 32*(3), 48–56.

Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE), 14*(2), 9.

Cesar, E., Cortés, A., Espinosa, A., Margalef, T., Moure, J. C., Sikora, A., et al. (2017). Introducing computational thinking, parallel programming and performance engineering in interdisciplinary studies. *Journal of Parallel and Distributed Computing, 105*, 116–126.

Chang, C. K. (2014). Effects of using Alice and Scratch in an introductory programming course for corrective instruction. *Journal of Educational Computing Research, 51*(2), 185–204.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education, 109*, 162–175.

Cheung, R. H. P. (2013). Exploring the use of the pedagogical framework for creative practice in preschool settings: A phenomenological approach. *Thinking Skills and Creativity, 10*, 133–142.

Choi, J., Lee, Y., & Lee, E. (2016). Puzzle based algorithm learning for cultivating computational thinking. *Wireless Personal Communications,* 1–15.

Computer Science Teachers Association (2011). CSTA K-12 computer science standards. Retrieve from: http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39.

Dillenbourg, P. (1999). *Collaborative learning: Cognitive and computational approaches. Advances in learning and instruction series.* PO Box 945, Madison Square Station, New York, NY 10160-0757: Elsevier Science, Inc.

Dodig-Crnkovic, G. (2011). Significance of models of computation, from Turing model to natural computation. *Minds and Machines, 21*(2), 301–322.

Erickson, H. L. (1998). *Concept-based curriculum and instruction: Teaching beyond the facts.* Thousand Oaks, CA: Corwin Press.

Evia, C., Sharp, M. R., & Pérez-Quiñones, M. A. (2015). Teaching structured authoring and DITA through rhetorical and computational thinking. *IEEE Transactions on Professional Communication, 58*(3), 328–343.

Falkner, K., Vivian, R., & Falkner, N. (2014, January). The Australian digital technologies curriculum: Challenge and opportunity. *Proceedings of the sixteenth Australasian computing education conference: 148*, (pp. 3–12). Australian Computer Society, Inc.

Farris, A. V., & Sengupta, P. (2016). Democratizing Children's Computation: Learning computational science as aesthetic experience. *Educational Theory, 66*(1–2), 279–296.

Fletcher, G. H., & Lu, J. J. (2009). Education human computing skills: Rethinking the K-12 experience. *Communications of the ACM, 52*(2), 23–25.

Forrest, S., & Mitchell, M. (2016). Adaptive computation: The multidisciplinary legacy of John H. Holland. *Communications of the ACM, 59*(8), 58–63.

de Freitas, E. (2016). Number sense and the calculating child: Measure, multiplicity and mathematical monsters. *Discourse: Studies in the Cultural Politics of Education, 37*(5), 650–661.

Grover, S., & Pea, R. (2013). Computational thinking in K–12 a review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237.

Gynnild, A. (2014). Journalism innovation leads to innovation journalism: The impact of computational exploration on changing mindsets. *Journalism, 15*(6), 713–730.

Heintz, F., Mannila, L., & Färnqvist, T. (2016, October). A review of models for introducing computational thinking, computer science and computing in K-12 education. *Frontiers in education conference (FIE), 2016 IEEE*(pp. 1–9). IEEE.

Hitchcock, C., Meyer, A., Rose, D., & Jackson, R. (2002). Providing new access to the general curriculum: Universal design for learning. *Teaching Exceptional Children, 35*(2), 8–17.

Hwang, G. J., & Tsai, C. C. (2011). Research trends in mobile and ubiquitous learning: A review of publications in selected journals from 2001 to 2010. *British Journal of Educational Technology, 42*(4), E65–E70.

Hwang, G. J., & Wu, P. H. (2014). Applications, impacts and trends of mobile technology-enhanced learning: A review of 2008-2012 publications in selected SSCI journals. *International Journal of Mobile Learning and Organisation, 8*(2), 83–95. http://dx.doi.org/10.1504/IJMLO.2014.062346.

Isbell, R. (2002). Telling and retelling stories – learning language and literacy. *Young Children, 57*(2), 26–30.

ISTE, C. (2011). *Computational thinking in K–12 education leadership toolkit.*

Jones, B. F., Rasmussen, C. M., & Moffitt, M. C. (1997). *Real-life problem solving: A collaborative approach to interdisciplinary learning.* American Psychological Association.

Kafai, Y. B., & Burke, Q. (2013). Computer programming goes back to school. *Phi Delta Kappan, 95*(1), 61–65.

Kameenui, E. J., & Carnine, D. W. (1998). *Effective teaching strategies that accommodate diverse learners.* Order Processing, PO Box 11071, Des Moines, IA 50336–1071: Prentice-Hall Inc.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). *Understanding computational thinking before Programming: Developing guidelines for the design.* Developments in Current Game-Based Learning Design and Deployment.

Kilpeläinen, P. (2010). Do all roads lead to Rome?(Or reductions for dummy travelers). *Computer Science Education, 20*(3), 181–199.

Kim, Y.-M., & Kim, J.-H. (2016). Application of a software education program developed to improve computational thinking in elementary school girls. *Indian Journal of Science and Technology, 9*(44).

Kokkos, A. (2010). Transformative learning through aesthetic experience: Towards a comprehensive method. *Journal of Transformative Education, 8*(3), 155–177.

Libeskind-Hadas, R., & Bush, E. (2013). A first course in computing with applications to biology. *Briefings in Bioinformatics, 14*(5), 610–617.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61.

Magana, A. J., & Silva Coutinho, G. (2017). Modeling and simulation practices for a computational thinking-enabled engineering workforce. *Computer Applications in Engineering Education, 25*(1), 62–78.
Manson, J. R., & Olsen, R. J. (2010). Diagnostics and rubrics for assessing learning across the computational science curriculum. *Journal of Computational Science, 1*(1), 55–61.
McCoy, J. D., & Ketterlin-Geller, L. R. (2004). Rethinking instructional delivery for diverse student populations: Serving all learners with concept-based instruction. *Intervention in School and Clinic, 40*(2), 88–95.
Navlakha, S., & Bar-Joseph, Z. (2011). Algorithms in nature: The convergence of systems biology and computational thinking. *Molecular Systems Biology, 7*(1), 546.
Ngan, S.-C., & Law, K. M. (2015). Exploratory Network Analysis of Learning Motivation Factors in e-Learning Facilitated Computer Programming Courses. *The Asia-Pacific Education Researcher, 24*(4), 705–717.
Orvalho, J. (2017, July). Computational thinking for teacher education. *Scratch2017BDX: Opening, inspiring, connecting* (pp. 6). .
Papert, S. (1990). *A critique of technocentrism in thinking about the school of the future. Epistemology and learning memo #2. September 1990.* Cambridge MA: MIT. Retrieved 29 August 2015 from www.papert.org/articles/ACritiqueofTechnocentrism.html.
Pellas, N., & Peroutseas, E. (2017). Leveraging Scratch4SL and second life to motivate high school students' participation in introductory programming courses: Findings from a case study. *New Review of Hypermedia and Multimedia, 23*(1), 51–79.
Repenning, A. (2012). Programming goes back to school. *Communications of the ACM, 55*(5), 38–40.
Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., ... Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education (TOCE), 15*(2), 11.
Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. *Computer supported collaborative learning* (pp. 69–97). Berlin, Heidelberg: Springer.
Rubinstein, A., & Chor, B. (2014). Computational thinking in life science education. *PLoS Computational Biology, 10*(11), e1003897.
Shell, D. F., & Soh, L. K. (2013). Profiles of motivated self-regulation in college computer science courses: Differences in major versus required non-major courses. *Journal of Science Education and Technology, 22*(6), 899–913.
Snodgrass, M. R., Israel, M., & Reese, G. C. (2016). Instructional supports for students with disabilities in K-5 computing: Findings from a cross-case analysis. *Computers & Education, 100*, 1–17.
Stefan, M. I., Gutlerner, J. L., Born, R. T., & Springer, M. (2015). The quantitative methods boot camp: Teaching quantitative thinking and computing skills to graduate students in the life sciences. *PLoS Computational Biology, 11*(4), e1004208.
Strickland, D. S., & Morrow, L. M. (1989). *Emerging literacy: Young children learn to read and write.* 800 Barksdale Rd., PO Box 8139, Newark, DE 19714–8139: International Reading Association.
Sysło, M. M., & Kwiatkowska, A. B. (2015, September). Introducing a new computer science curriculum for all school levels in Poland. *International conference on informatics in Schools: Situation, evolution, and perspectives* (pp. 141–154). Cham: Springer.
Wang, H. Y., Liu, G. Z., & Hwang, G. J. (2017). Integrating socio-cultural contexts and location-based systems for ubiquitous language learning in museums: A state of the art review of 2009–2014. *British Journal of Educational Technology, 48*(2), 653–671.
Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.
Wilkerson-Jerde, M. H. (2014). Construction, categorization, and consensus: Student generated computational artifacts as a context for disciplinary reflection. *Educational Technology Research and Development, 62*(1), 99–121.
Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.
Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London a: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725.
Wing, J. M. (2010). Computational thinking: What and why? Retrieved from http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf.
Wolz, U., Stone, M., Pearson, K., Pulimood, S. M., & Switzer, M. (2011). Computational thinking and expository writing in the middle school. *ACM Transactions on Computing Education (TOCE), 11*(2), 9.
Wood, D. F. (2003). ABC of learning and teaching in medicine: Problem based learning. *BMJ: British Medical Journal, 326*(7384), 328.
Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE), 14*(1), 5.
Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research, 53*(4), 562–590.