AISpace2: An Interactive Visualization Tool for Learning and Teaching Artificial Intelligence

Chenliang Zhou,[1] Dominic Kuang, Jingru Liu, Hanbo Yang, Zijia Zhang, Alan Mackworth, David Poole[2]
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada
[1]chenliang.zhou@qq.com
[2]poole@cs.ubc.ca

Abstract

AIspace is a set of tools used to learn and teach fundamental AI algorithms. The original version of AIspace was written in Java. There was not a clean separation of the algorithms and visualization; it was too complicated for students to modify the underlying algorithms. Its next generation, AIspace2, is built on AIPython, open source Python code that is designed to be as close as possible to pseudocode. AISpace2, visualized in JupyterLab, keeps the simple Python code, and uses hooks in AIPython to allow visualization of the algorithms. This allows students to see and modify the high-level algorithms in Python, and to visualize the output in a graphical form, aiming to better help them to build confidence and comfort in AI concepts and algorithms. So far we have tools for search, constraint satisfaction problems (CSP), planning and Bayesian network. In this paper we outline the tools and give some evaluations based on user feedback.

Introduction

Artificial intelligence (AI) is becoming a more and more popular and important field in computer science. However, both teaching and learning AI are not simple for many instructors and students, partly because the concepts and algorithms are more abstract and not as straightforward as the ones the students encountered before in a basic algorithm course (Greiner and Schaeffer 2001; Hearst 1994).

Several researchers have pointed out that the use of certain level of multimedia, such as animation and visualization, in presenting course materials is more effective to capture the attention from students and thus more helpful in motivating their interest to learn AI (Naps et al. 2003; Demetriadis, Triantfillou, and Pombortsis 2003). Compared to traditional paper-based learning, strategies that involve computers could help decrease students' cognitive load (Khalil, Mansour, and Wilhite 2010) and its outcome is at least as good as the traditional teaching strategies (Valentine, Belski, and Hamilton 2017). It is also shown in Zeng's work (2016) that computer visualization can promote the understanding of models in deep learning, a subfield of AI. There is evidence that those visualized AI algorithms, although complicated, could become more accessible and less intimidating to students (Kehoe, Stasko, and Taylor 2001). A more specific experiment has been done to show that the visualization tools can help students learn and understand search AI algorithms (Naser 2008). Although some experiments have been conducted to show that animations in learning may bring a fake feeling of satisfaction, negatively impacting the development of learning strategies, they are impressively advantageous for learners to understand dynamic activities, and can give them more confidence and motivation (Ainsworth 2008). Therefore, to engage students and encourage them to learn more actively, we built tools so the execution of the algorithms can be visualized and students can interact with and control these visualizations.

In 2016, we started a project called AISpace2 (https://aispace2.github.io/AISpace2/), an interactive and visualizable pedagogical tool designated to help learn and teach AI knowledge. In this paper we give an introduction to AISpace2. We first present some background information about CIspace and AIspace, the previous generations of AISpace2, since their first launch in 1999 (Poole and Mackworth 2001; Knoll et al. 2008), and then AIPython (Poole and Mackworth 2017a), on which AISpace2 is built. We then talk about the design approaches adopted in AISpace2 and their advantages compared to its predecessors. Next, we outline some important features in AISpace2 followed by some evaluations of our project based on user feedback. Finally, we give a plan of our future work, including a summary of possible improvements and achievements we could make.

Related Work

In this section we briefly talk about the history of CIspace (Poole and Mackworth 2001), AIspace (Knoll et al. 2008) and AIPython.

CIspace

In 1999, Poole and Mackworth launched CIspace (2001), a set of nine visualization applets aimed at helping to learn AI concepts and algorithms interactively based on their textbook *Computational Intelligence: A Logical Approach* (1998). The goal of CIspace is to enhance approaches to learning and teaching AI, which can be further decomposed into the usability goal and the pedagogical goal (Amershi et

al. 2005a). CIspace has been incorporated into undergraduate courses at many universities and received positive feedback from students and instructors (Amershi et al. 2005b). Experiments show that the CSP applet in CIspace is at least as good a study tool as the traditional paper-based study, and that students prefer to study using the applet rather than with traditional paper-based methods (Amershi et al. 2005b).

AIspace

In 2008, CIspace was renamed as AIspace and its developers continued on actively developing and releasing new versions with more and more new features added (Knoll et al. 2008). Figure 1 shows its applet for CSP. AIspace is more closely aligned with the textbook of Poole and Mackworth, *Artificial Intelligence: Foundations of Computational Agents* (2017b). In AIspace, customizable applets were introduced, making the tools easier to use for different levels of users, and helping instructors for online tutorials of a specific AI concept (Knoll et al. 2008). An evaluation of AIspace was made by surveying students questions about effectiveness, usability and enjoyability of the tools and the average score in the feedback was 3.8 out of 5 (Knoll et al. 2008). This is a desirable result but there is still room for improvement. One possible improvement is to provide adaptive support, whose effectiveness is shown in Kardan and Conati's work (2015).

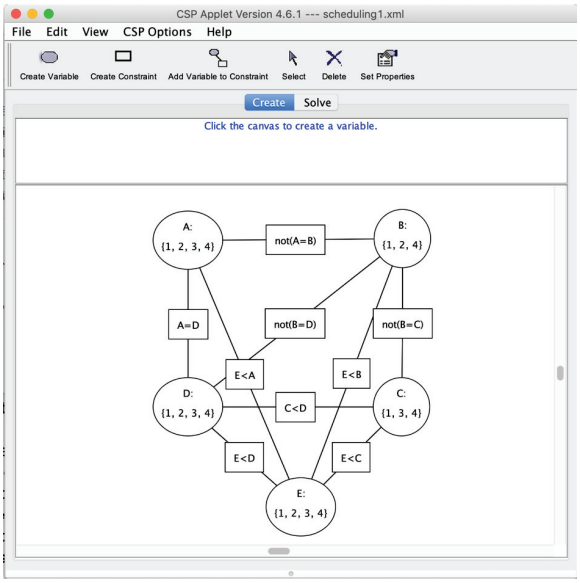
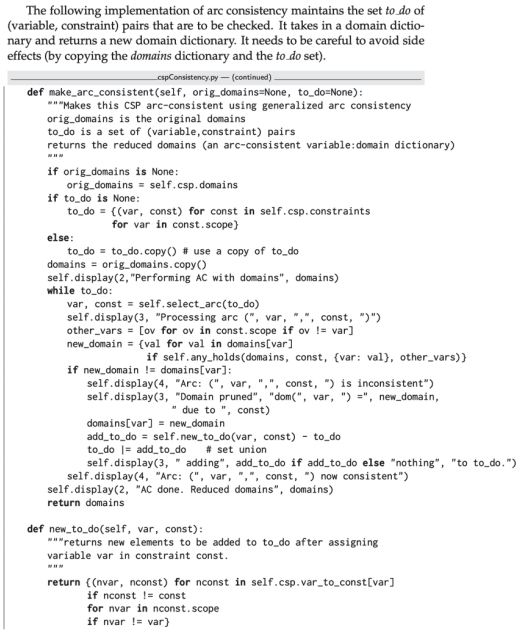


Figure 1: AIspace CSP Applet

AIPython

AIPython (http://aipython.org) is a high-level implementation of most of the algorithms found in Poole and Mackworth's *Artificial Intelligence: Foundations of Computational Agents* (2017b). It is designed to be as close as possible to pseudocode. AIPython tries to have a good asymptotic complexity, but given a design choice, clarity is preferred over efficiency. For example, if a set is the correct abstraction, it uses the `set` in Python, even if the set is small and it probably would have been more efficient to use the `list`. AIPython is designed to be able to run independently in Python 3 with a minimal number of libraries; it enables students to use it without needing to learn external libraries. Figure 2 is a page from the code documentation showing the arc consistency algorithm in AIPython.


The following implementation of arc consistency maintains the set *to_do* of (variable, constraint) pairs that are to be checked. It takes in a domain dictionary and returns a new domain dictionary. It needs to be careful to avoid side effects (by copying the *domains* dictionary and the *to_do* set).

cspConsistency.py — (continued)

```
24  def make_arc_consistent(self, orig_domains=None, to_do=None):
25      """Makes this CSP arc-consistent using generalized arc consistency
26      orig_domains is the original domains
27      to_do is a set of (variable,constraint) pairs
28      returns the reduced domains (an arc-consistent variable:domain dictionary)
29      """
30      if orig_domains is None:
31          orig_domains = self.csp.domains
32      if to_do is None:
33          to_do = {(var, const) for const in self.csp.constraints
34                   for var in const.scope}
35      else:
36          to_do = to_do.copy() # use a copy of to_do
37      domains = orig_domains.copy()
38      self.display(2,"Performing AC with domains", domains)
39      while to_do:
40          var, const = self.select_arc(to_do)
41          self.display(3, "Processing arc (", var, ",", const, ")")
42          other_vars = [ov for ov in const.scope if ov != var]
43          new_domain = {val for val in domains[var]
44                        if self.any_holds(domains, const, {var: val}, other_vars)}
45          if new_domain != domains[var]:
46              self.display(4, "Arc: (", var, ",", const, ") is inconsistent")
47              self.display(3, "Domain pruned", "dom(", var, ") =", new_domain,
48                           " due to ", const)
49              domains[var] = new_domain
50              add_to_do = self.new_to_do(var, const) - to_do
51              to_do |= add_to_do    # set union
52              self.display(3, " adding", add_to_do if add_to_do else "nothing", "to to_do.")
53          self.display(4, "Arc: (", var, ",", const, ") now consistent")
54      self.display(2, "AC done. Reduced domains", domains)
55      return domains
56
57  def new_to_do(self, var, const):
58      """returns new elements to be added to to_do after assigning
59      variable var in constraint const.
60      """
61      return {(nvar, nconst) for nconst in self.csp.var_to_const[var]
62              if nconst != const
63              for nvar in nconst.scope
64              if nvar != var}
```

Figure 2: Arc Consistency Algorithm in AIPython

AIPython uses a method `display(level, *args, **kwargs)` to trace the code (where the level is a threshold for the amount of detail a user wants), and a `select` for making selections that do not affect the correctness, but may affect the efficiency of code. When run in Python, these just print or select an arbitrary value. These functions provide hooks that can be used in a visualization or to let a user select values. AISpace2 does not modify the code, but redefines `display` and `select` to allow for interactive visualization of the code.

Design Approaches

Our major goal for AISpace2 is to make the tools in AIspace more accessible and extendable, and to enable and encourage students to actively interact and investigate more deeply, not only with the visualization, but also with the AIPython source code in the backend, so that they can understand more thoroughly the mechanism of each algorithm.

Figure 3 illustrates the overall structure of AISpace2. We use JupyterLab (Granger and Grout 2016) widgets to represent our visualizations, which are driven by Vue.js (You et al. 2014) renderers. We use Python files as the backend and use AIPython as core algorithms. We also use TypeScript (Microsoft 2012) to handle the frontend events. Additionally, we have several Python files serving as bridges to enable the communication between frontend and backend.

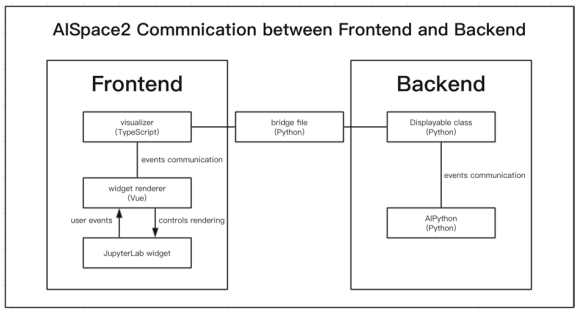


Figure 3: AISpace2 Overall Structure

Visualizations in JupyterLab

In order to make AISpace2 more accessible and extendable, we chose to use JupyterLab (Granger and Grout 2016) widgets to present our frontend visualizations instead of the downloadable Java applets used in AIspace. This is because Jupyter (Perez and Granger 2015) provides an environment for open-source software, especially popular in the fields of machine learning and artificial intelligence, and services for interactive computing across dozens of programming languages.

JupyterLab is the next generation of the Jupyter Notebook interface. The notebook document format (Kluyver et al. 2016) provided by Jupyter combines explanatory text, mathematics, computations and their rich media output and can be directly run in a browser. This makes the use of AISpace2 more convenient and accessible compared to its previous versions. Users can edit the code and test it directly in the browser, save their changes in the notebook file and share it with others, making it extremely extendable. A problem with Jupyter Notebook is that all code has to be in one sequence and input files are hidden. JupyterLab provides a side file manager panel and allows user to open multiple file tabs at once. Since we write solvers for different kinds of problems in separate notebook files, these features of a side file panel and file tabs make it more convenient for users to switch between different solvers and algorithms and make comparisons, and to edit files that are loaded by the notebook.

Incorporate AIPython as Backend

In order to enable and encourage students to actively interact and investigate more deeply, we incorporated AIPython into AISpace2.

We decided to expose the AIPython code to users because we think there might be many students who find the frontend visualization too magical and wish to go deeper to investigate what is really going on in the underlying algorithms. By exploring the backend, we expect that students would gain more enhanced knowledge about the concepts and algorithms described by the frontend visualization. Furthermore, users are allowed to modify the backend code because in many situations students may be curious and asking "what will happen if ..." and want to try themselves. We expect that users can gain more solid knowledge by modifying and experimenting with the backend AIPython code.

In order to let AIPython, which is designed to run independently, cooperate better with the frontend, the only modification to the AIPython code is a `@visualize` decorator, which wraps the functions that are to be directly called in JupyterLab. The `@visualize` decorator makes those functions automatically run when the JupyterLab widget is rendered and does not make the code less readable. Adding this decorator is necessary because we need to tell the frontend which functions are driving the visualization, sending messages to it and running in parallel with it; otherwise these functions will block the main thread.

Connect Frontend and Backend

We used TypeScript (Microsoft 2012) and Vue.js (You et al. 2014) as the frontend languages. In order to connect the frontend and backend, we redefined the `Displayable` classes, redefining `display` and `select`, sending signals to the frontend about which functions are to be visualized and how they should be visualized. For each tool (e.g. search, CSP, planning, etc.), we implemented a frontend visualizer and a backend `Displayable` class. Combined together, they are responsible for handling the event communication between the frontend and the backend. This includes both the controlling events and rendering events. The controlling events are those sent from frontend to backend, triggered when, for example, the user clicks on the Solve button or a specific node in the JupyterLab widget. The rendering events are those sent back from the backend to the frontend and are related to the tasks of graph rendering and message outputting. They are triggered whenever there is a graph changing or some printout needing to be presented on the screen.

In order to make the two languages (TypeScript in the frontend visualizer and Python in the backend AIPython) communicate with each other successfully and effectively, we set up bridges between them. For each tool we have a corresponding bridge file, responsible for converting a Python representation of a problem, received from backend, to its JSON representation and sending it to the frontend visualizer, incorporating the necessary information to render it as a graph, and also reversely, converting JSON to Python.

Features

In this section we outline the main features of AISpace2, including its problem solvers, controls and problem builders.

Solvers

The solvers are the main tools of AISpace2. For each algorithm we have an associated solver. Figure 4 shows a CSP solver using the arc consistency code presented in Figure 2. Because for each kind of problem, there might be multiple algorithms to solve it, each of which has its own strengths and weaknesses, AIPython provides the support for multiple algorithms. For example, in search problems, it provide solvers for depth-first search, breadth-first search, A* search, branch-and-bound search, etc., and in CSP, it

provides solvers for the arc consistency algorithm and for stochastic local search algorithms.

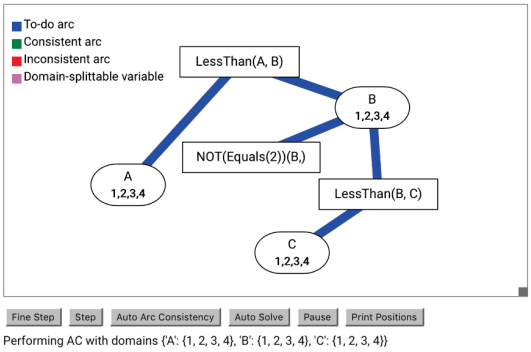


Figure 4: A CSP Solver Using Arc Consistency Algorithm

Students are able to switch between different solvers by switching between the notebooks, which is made extremely smooth thanks to the side file panel and file tab functions in JupyterLab (Granger and Grout 2016). With two tabs open simultaneously, it is also easier for students to recognize certain subtle difference between similar algorithms (e.g. forward planning and regression planning), and to compare the effects of different kinds of problems on the performance of these algorithms.

Controls

The controls for the visualization are designed to ensure that it is easy to use. The controls can be classified into two categories: main controls and visualization controls.

For all of the solvers, the main interface through which users can control the visualization is by clicking on nodes and arcs or using one of the buttons `Fine Step`, `Step` or `Auto Solve`, located at the bottom of each visualization widget. These buttons provide different granularities of solving a problem to enable different demands from users; for example, it is easy to imagine the huge gap in the level of proficiency in the algorithms among users: some students who just started to learn the material might have little understanding of the algorithms and they may wish to step into every detail happening during the process of the algorithm, while other students or instructors already proficient in the algorithm may just wish to see an overview of the execution or a final answer. By providing different granularities we can satisfy a wide range of users.

Two other convenient functions are `Pause` and `Print Positions` buttons. The `Pause` button can pause the execution of an algorithm between each fine-step during a step or an auto-solve, providing an opportunity for users to investigate the intermediate state of the problem. The `Print Positions` button prints out the position information about the objects in the graph, which can be utilized by the users when they want to construct a new problem or modify an existing problem but want the nodes to locate at the same position when rendered.

Apart from the main controls, there are also various visualization controls. Some of the important ones are summarized in Table 1. These controls are mainly about different visualization options and can be accessed through Python. Through tuning these parameters, users can create a visualization they desire. Although these controls are convenient, we choose to hide them from the notebook interface and users have to go to `Displayable` classes in Python to change them because these options are less related to the AI algorithms and concepts themselves and we do not want to distract users from what they should focus on. We expect that normal users will not need to worry about these visualization controls and use the default setup for them. For example, in an A* search solver, the default setup is `show_edge_costs=True` and `show_node_heuristics=True` because in most cases the users want to observe the edge costs and node heuristics in the execution of the A* algorithm.

Builders

For each category of problems (search, CSP, etc.), we provide several predefined sample problems that the user can load. If the user wants to build a problem themselves, they can write the Python code to construct instances of the corresponding problem classes. However, we noticed that some students are not familiar with the Python programming language and the details of the representations and may find these difficult, which is undesirable because the major goal of AISpace2 is to facilitate learning and understanding of important AI concepts and algorithms instead of learning how to write code in Python. Therefore, for each kind of problem, we provide a problem builder to help users construct problems visually, taking advantage of the fact that the problems we are dealing with can be represented in the form of a graph. Figure 5 shows a Bayesian network builder, where the user is setting the conditional probability table for the variable `Grass wet`.

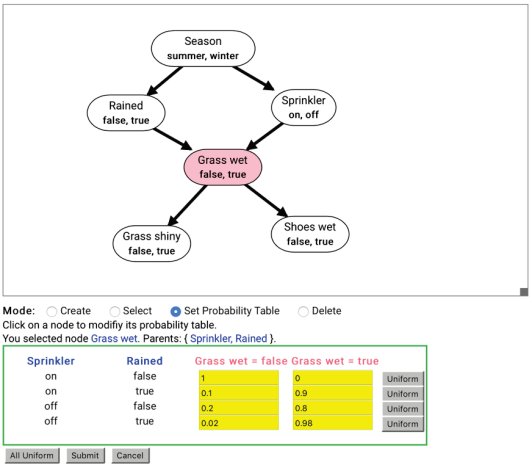


Figure 5: Bayesian Network Builder

A user can either choose to create their own problem from an empty graph, or choose to build upon an existing problem. In the builders, the user can change the graph by simply clicking or dragging a node or an edge on the screen.

Table 1: Examples of Visualization Controls

Control	Meaning
sleep_time	The waiting time between each step in auto solving
line_width	The thickness of edges
text_size	The font size of the text
detail_level	The detail level of the information shown on a node
show_edge_costs	Determines whether to show the cost of each edge (only in search related problems)
show_node_heuristics	Determines whether to show the heuristic value inside each node (only in search related problems)
decimal_place	The decimal place of the query result (only in probability related problems)

They can also perform various operations on the graph including adding or deleting a node or an edge, and changing the attributes (e.g. name) of a node or an edge. Of course, there are some functions exclusive to each kind of problem. For example, in a search problem builder the user can set the heuristic value of a node or the cost of an edge, and in a Bayesian network builder the user can set the probability table of a variable, which is represented as a node in the graph. Once they have created the problem they can use the provided solvers on it.

Evaluation

AISpace2 is a pedagogical tool, so it is essential that it is user-friendly and effective in helping users understand the underlying AI concepts and algorithms. In order to evaluate the pedagogical effectiveness and usability of AISpace2, we performed two evaluations on both experienced and inexperienced AI learners.

Experiment One: Controlled Experiment on CSP Study

One of the main goals of AISpace2 is to help students understand the AI concepts and algorithms clearly. Hence we conducted a controlled experiment on knowledge gain of a specific AI algorithm, the arc consistency algorithm for CSPs, to evaluate the pedagogical effectiveness of AISpace2. This experiment also collected some user feedback about using AISpace2, helping evaluate the usability of AISpace2 to some extent.

20 voluntary undergraduate students from universities in Canada and China who have never taken AI courses were invited to participate in this experiment. We first randomly classified them into two groups: AISpace2 group and non-AISpace2 group, each of ten. Then we provided them with textbook material that contains the required knowledge about CSP and arc consistency algorithm (sections 4.1-4.5 in *Artificial Intelligence: Foundations of Computational Agents* (Poole and Mackworth 2017b)). In addition, students in the AISpace2 group were provided with our AISpace2 CSP tool (they were first instructed about the installation and the use of AISpace2) when learning the required textbook material, whereas students in the non-AISpace2 group, as a control group, were not provided with extra materials.

All 20 students were then asked to complete the quiz of the same content testing on five concepts about CSP and arc consistency algorithm: variable, constraint, constraint network, arc consistency and domain splitting (Figure 6). The quiz contained six questions, each worth one mark, totaling a full mark of six. We did not allow the use of AISpace2 during the quiz as this would trivialize the quiz questions. We recorded the average scores for each group.

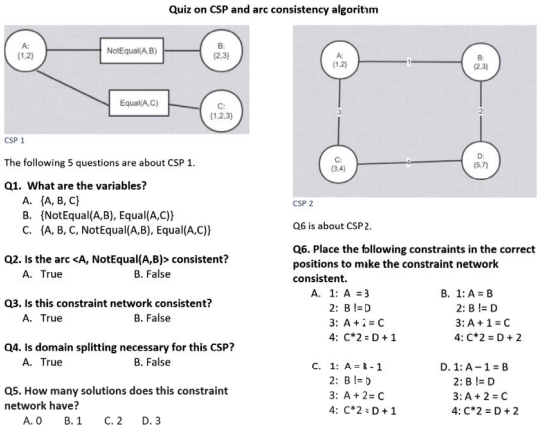


Figure 6: CSP Quiz for Experiment One

After the quiz, they filled in a self-assessment sheet where they rated their understanding of the corresponding concepts on a five-level Likert scale, with a rating of 1 meaning that they completely did not understand and a rating of 5 meaning that they completely understood. We recorded the average self-assessment score for each group in Table 2. Finally, the participants in the AISpace2 group were asked about the time they spent in learning how to use AISpace2, and all participants were asked about their study preferences. The options for study preference for the AISpace2 group were "textbook only," "AISpace2 only" and "textbook with AISpace2," whereas the options for the the non-AISpace2 group were "textbook only" and "textbook with other study aid." The results are shown in Figure 7.

For the quiz score, the median was 4 for both groups. However, the average score for the AISpace2 group (4.6) was notably higher than the average score for the non-AISpace2 group (3.9). Furthermore, we can see from Table 2 that students in the AISpace2 group seemed to have a better understanding of all five concepts in CSP and arc consistency algorithm than the non-AISpace2 group. This might indicate that study with AIspace2 was more effective in helping them understand AI concepts, at least those about CSP and arc consistency algorithm, than the tradi-

Table 2: Average Self-Assessment Score on Five Concepts in CSP

Group	Variables	Constraints	Constraint network	Arc consistency	Domain splitting
AISpace2 group	4.7	4.7	4.8	4.6	4.5
Non-AISpace2 group	4.6	4.4	4.2	4.2	4.0

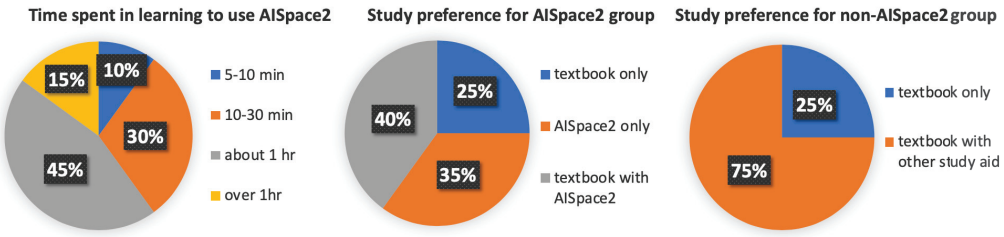


Figure 7: Pie Charts for Feedback on AISpace2 and Quiz

tional way of textbook only. We also noticed that for relatively abstruse concepts such as constraint networks, arc consistency and domain splitting, the difference between average self-assessment score was even greater. For example, the average self-assessment score for the simple concept of variables in the AISpace2 group was only 0.1 higher than the non-AISpace2 group, but this gap increased to 0.5 for the concept of domain splitting. This suggests that for basic concepts, textbook alone may suffice, but when it comes to the abstruse one, it would probably be better to equip learners with some other study aid to help digest the knowledge.

Moreover, we can see from Figure 7 that in the non-AISpace2 group, about three quarters of the participants preferred to study with some other aid along with a textbook, and this is where we hypothesize that AISpace2 could play the crucial role. However, about 60 percent of students spent one hour or more in getting themselves familiar with AISpace2, which is slightly more than we expected. This might suggest that we need a more detailed and accessible tutorial to guide the users.

Experiment Two: Student Evaluation of AISpace2 Tools

As convenience, accessibility and extensibility are among the major concerns of AISpace2 design, in the second experiment we asked undergraduate students who have taken an AI course that used AISpace to compare the effectiveness of AISpace2 and AIspace.

The aim of this is to evaluate the improvements, if any, in AISpace2 over AIspace and also draw a brief picture to see the potential deficiencies to address in the future.

16 undergraduate students participated in this experiment. They were first instructed about the installation and the use of AISpace2. They rated AISpace2 experience compared to their previous experience on AIspace on four categories of problems (search, CSP, planning and Bayesian network) in two indices (effectiveness and convenience) on a five-level Likert scale, where a rating of 1 means the experience was much poorer than AIspace experience and a rating of 5 means much better.

The result is recorded in Table 3. We then also asked and recorded whether they would use AISpace2 in place of AIspace as a study aid tool. The participants could respond on a five-level Likert scale, where a rating of 1 means they definitely would not and a rating of 5 means they definitely would.

We can see from Table 3 that the overall feedback received from participants was positive. The grand average was 4.6 out of 5, which means that in general our users thought that their AISpace2 experience was notably better than AIspace experience. The average rating for the effectiveness index was also 4.6 and that for the convenience index was slightly lower (4.5). We can see from the participants' comments that this may be partly due to the "complicated installation process" as opposed to a simple download for AIspace. Despite the installation, the other comments were mostly encouraging, which was consistent with the participants' average rating of 4.6.

When looking at the different categories of problems from the table, we can see that AISpace2 received most approval for Bayesian networks (4.7), which benefited from its highest average in convenience index (4.8), 0.3 higher than the average convenience index among all categories. This reveals that to improve AISpace in the future work, it might be beneficial to analyze the tools for Bayesian networks. It should also be brought to the attention that our tools for planning problem received the lowest mark in both effectiveness (4.4) and convenience indices (4.3); this might be attributed to, as pointed out by some participants, the slow performance of visualization when they transformed a planning problem, represented in the form of Stanford Research Institute Problem Solver (STRIPS) (Fikes and Nilsson 1971), to a CSP and set the horizon (the number of actions expected to finish the planning task) to a high value because this transformation would create a CSP with a huge constraint network. This issue might also happen in other solvers when there are too many nodes or arcs in the graph, and we would work on this in the future to seek a way to optimize the per-

Table 3: Ratings on AISpace2 Experience Compared to AIspace

Index	Search	CSP	Planning	Bayesian	Summary
Effectiveness	μ=4.8 σ=0.2 n=16	μ=4.5 σ=0.3 n=16	μ=4.4 σ=0.2 n=16	μ=4.6 σ=0.2 n=16	μ=4.6 σ=0.1 n=64
Convenience	μ=4.4 σ=0.5 n=16	μ=4.6 σ=0.2 n=16	μ=4.3 σ=0.3 n=16	μ=4.8 σ=0.2 n=16	μ=4.5 σ=0.6 n=64
Sum	μ=4.6 σ=0.4 n=32	μ=4.6 σ=0.3 n=32	μ=4.4 σ=0.3 n=32	μ=4.7 σ=0.2 n=32	μ=4.6 σ=0.3 n=128

Note: μ=average, σ=standard deviation, n=sample size

formance in this situation.

For their willingness to choose AISpace2 in place of AIspace, 50 percent of students rated 4, meaning that they would probably consider using AIspace2 as an aid tool instead of AIspace, and the rest 50 percent rated 5, meaning that they definitely would choose AISpace2. This result implies that the vast majority of students would think there are significant improvements in AISpace2, which to some extent verifies its pedagogical effectiveness and convenience, at least in the scope of materials used in the introductory AI course. This indicates that we could be ready to launch AISpace2 in real classes in the near future.

Summary

The two experiments examined performance of AISpace2 on students with or without experience in AI, revealing a relatively comprehensive assessment. Experiment One showed that AISpace2 provides a powerful and effective assistance in learning concepts of CSP and arc consistency algorithm and students are willing to use it accompanying the textbook. The self-assessment scores also give an encouraging result. However, as noted in the work of Ainsworth (2008), the use of the animations could bring a fake feeling of satisfaction hence a biased assessment of confidence. Therefore, the effectiveness of AISpace2 as a pedagogical tool still needs to be supported by more empirical evidence given by a long-term assessment such as its performance on assignments and exams.

Experiment Two collected feedback on AISpace2 from students already having some experience in AI. We can see that the feedback is highly positive (average 4.6 out of 5) and this is also consistent with the results of Experiment One.

Future

In the future, in addition to continuous maintenance, our focus would be placed on two major aspects: improvement and innovation. As mentioned in the evaluation section, although AISpace2 received positive feedback, long-term examination and assessment are necessary. Therefore, our first goal is to launch AISpace2 in real classes. Based on feedback from students and instructors such as students' performance on assignments and exams, we should be able to find where we need to improve. Also, as noted in the evaluation section, we may need to simplify the installation process more and to optimize the visualization performance when the graph is very big. We are also considering launching AISpace2 on JupyterHub (Milligan 2017), a customizable and flexible platform using JupyterLab but requiring a simpler process of installation, making the use of AISpace2 more convenient.

The second major goal in the near future is to introduce new features to AISpace2. Currently, AISpace2 supports various algorithms in search, CSPs, planning and Bayesian networks. Apart from these basic topics, we plan to add more advanced ones such as neural networks, deep learning, Markov models, etc.

AISpace2 is a pedagogical tool aimed at assisting AI learners to visually comprehend AI concepts, models and algorithms. It is based on its previous generation, AIspace, but aims to improve the usability and effectiveness, and will be continually developed with new features. The usability and effectiveness of AISpace2 has been qualitatively investigated in our two experiments, and we believe that after its launch in real classes in the near future, the empirical results could provide more constructive advice. As AI is becoming an indispensable field of computer science, we foresee that AISpace2, designed as an effective learning tool to help understand AI knowledge, will benefit many students and teachers.

Acknowledgments

The AISpace2 project has been financially supported by the University of British Columbia (UBC) through its Science Undergraduate Research Experience Award (SURE) and by the Natural Sciences and Engineering Research Council of Canada (NSERC) through its Undergraduate Student Research Awards (USRA) and discovery grants. The AISpace2 project has been designed, developed and maintained by teams of students and faculty in the Department of Computer Science at UBC. We sincerely thank the AISpace2 contributors including Richard Chiang and Anna Zheltukhina. We also thank the anonymous participants of the two experiments to help our evaluations.

References

Ainsworth, S. 2008. How Do Animations Influence Learning. *Current Perspectives on Cognition, Learning, and Instruction: Recent Innovations in Educational Technology That Facilitate Student Learning* 37–67.

Amershi, S.; Arksey, N.; Carenini, G.; Conati, C.; Mackworth, A.; Maclaren, H.; and Poole, D. 2005a. Designing CIspace: Pedagogy and Usability in a Learning Environment for AI. *ACM SIGCSE Bulletin* 37(3):178–182.

Amershi, S.; Arksey, N.; Carenini, G.; Conati, C.; Mackworth, A.; Maclaren, H.; and Poole, D. 2005b. Fostering Student Learning and Motivation: An Interactive Educational Tool for AI. Technical report, Citeseer.

Demetriadis, S.; Triantfillou, E.; and Pombortsis, A. 2003. A Phenomenographic Study of Students' Attitudes Toward the Use of Multiple Media for Learning. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '03, 183–187. New York, NY, USA: ACM.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2(3-4):189–208.

Granger, B., and Grout, J. 2016. JupyterLab: Building Blocks for Interactive Computing. *Slides of Presentation Made at SciPy* 2016.

Greiner, R., and Schaeffer, J. 2001. AIxploratorium: A Vision for AI and the Web. In *In Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Effective Interactive AI Resources*. Citeseer.

Hearst, M. A. 1994. Preface: Improving Instruction of Introductory AI. In *Improving Instruction of Introductory Artificial Intelligence: Papers from the AAAI Fall Symposium*.

Kardan, S., and Conati, C. 2015. Providing Adaptive Support in an Interactive Simulation for Learning: An Experimental Evaluation. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 3671–3680. ACM.

Kehoe, C.; Stasko, J.; and Taylor, A. 2001. Rethinking the Evaluation of Algorithm Animations as Learning Aids: An Observational Study. *International Journal of Human-Computer Studies* 54(2):265–284.

Khalil, M. K.; Mansour, M. M.; and Wilhite, D. R. 2010. Evaluation of Cognitive Loads Imposed by Traditional Paper-Based and Innovative Computer-Based Instructional Strategies. *Journal of Veterinary Medical Education* 37(4):353–357.

Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B. E.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J. B.; Grout, J.; Corlay, S.; et al. 2016. Jupyter Notebooks - a Publishing Format for Reproducible Computational Workflows. In *ELPUB*, 87–90.

Knoll, B.; Kisynski, J.; Carenini, G.; Conati, C.; Mackworth, A.; and Poole, D. 2008. AIspace: Interactive Tools for Learning Artificial Intelligence. In *Proceedings of the AAAI 2008 AI Education Workshop*, 3.

Microsoft. 2012. TypeScript. https://www.typescriptlang.org.

Milligan, M. 2017. Interactive hpc gateways with jupyter and jupyterhub. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, 63. ACM.

Naps, T.; Cooper, S.; Koldehofe, B.; Leska, C.; Rößling, G.; Dann, W.; Korhonen, A.; Malmi, L.; Rantakokko, J.; Ross, R. J.; Anderson, J.; Fleischer, R.; Kuittinen, M.; and McNally, M. 2003. Evaluating the Educational Impact of Visualization. *ACM SIGCSE Bulletin* 35(4):124–136.

Naser, S. S. A. 2008. Developing Visualization Tool for Teaching AI Searching Algorithms. *Information Technology Journal, Scialert* 7(2):350–355.

Perez, F., and Granger, B. E. 2015. Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science. *Retrieved September* 11(207):108.

Poole, D. L., and Mackworth, A. K. 2001. CIspace: Tools for Learning Computational Intelligence. In *Proceedings of the Workshop on Effective Interactive AI Resources*.

Poole, D., and Mackworth, A. 2017a. AIPython: Python Code for Artificial Intelligence: Foundations of Computational Agents. aipython.org.

Poole, D. L., and Mackworth, A. K. 2017b. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press.

Poole, D.; Mackworth, A.; and Goebel, R. 1998. *Computational Intelligence: A Logical Approach*. Oxford University Press.

Valentine, A.; Belski, I.; and Hamilton, M. 2017. Developing Creativity and Problem-Solving Skills of Engineering Students: A Comparison of Web- and Pen-and-Paper-Based Approaches. *European Journal of Engineering Education* 42(6):1309–1329.

You, E.; Kadyan, R.; Wedrychowski, D. G.; Drasner, S.; Wu, P.; Eduardo; An, P.; et al. 2014. Vue.js. https://vuejs.org.

Zeng, H. 2016. Towards Better Understanding of Deep Learning with Visualization. *The Hong Kong University of Science and Technology*.