Creative Invention Benchmark

Matthew Guzdial, Nicholas Liao, Vishwa Shah, and Mark O. Riedl
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332 USA
mguzdial3@gatech.edu, nliao7@gatech.edu, vishwashah@gatech.edu, riedl@cc.gatech.edu

Abstract

In this paper we present the Creative Invention Benchmark (CrIB), a 2000-problem benchmark for evaluating a particular facet of computational creativity. Specifically, we address combinational p-creativity, the creativity at play when someone combines existing knowledge to achieve a solution novel to that individual. We present generation strategies for the five problem categories of the benchmark and a set of initial baselines.

Introduction

Benchmarks represent a common means for driving community effort on a specific task. For example, MNIST is a dataset of handwritten digits paired with their numeric value, which proved popular and produced breakthroughs in the field of image recognition (LeCun 1998). At present it is considered solved by modern methods, but has continued as a standard for evaluating novel approaches, given that there are known performances for comparable techniques. While imperfect, we posit a benchmark for creativity could accomplish similar effects for the field of computational creativity. Creativity itself is too ill-defined for any benchmark. But just as recognizing handwritten digits does not translate to mastery of computer vision, we can define a set of creative tasks to address one particular facet of creativity.

Imagine you are an agent with a limited knowledge base. You know about the color red (255,0,0) and blue (0,0,255), and you know that there are integer variables x and y that can range from 0 to 100. You have access to a single method **Paint**, that given a value for x and y, and a color (represented in RGB) paints a pixel of a canvas. In return for painting a pixel the agent receives a floating point score [0-1] that grades the agents current painting compared to an unseen goal. The goal, in this case, is to paint a picture of a grape.

As a human reading the problem description above, the answer to this problem appears obvious. From red and blue make purple, or perhaps multiple shades of purple, in order to paint the unseen picture of a grape. This instinct can be understood as an instantiation of what Boden calls combinational creativity (Boden 2004). But it does not reflect how a naive AI agent might solve this problem, instead greedily placing blue and red to maximize the score to some local maximum without inventing the color purple. Alternatively one might naively hand-author all possible colors for the AI agent, but this would run against the spirit of the problem. Solving this problem clearly requires creativity, but we do not argue that this problem can evaluate the entirety of creativity. We instead focus on *p-creative*, *combinational* creativity (Boden 1998). P-creativity refers to creation of artifacts that are novel to the individual creator based on its knowledge (e.g., the artifact could have been invented by other creators previously). Combinational creativity refers to the creation of artifacts through the process of recombining existing knowledge. For the purposes of this paper we refer to this class of problem as invention problems.

In this paper we present the Creative Invention Benchmark (CrIB)[1], a publicly available benchmark of 2000 problems in 5 domains (painting, alien language, photobashing, narrative, and dessert recipes). All of these problems fit the general form of the painting example, requiring an agent to generalize and invent new concepts from a given problem-specific knowledge base to reach a solution given feedback from an unseen goal.

The example of painting an unseen grape may seem trivial but it is analogous to many of the most interesting and practical problems currently facing society from product invention to drug discovery. As humans we reflect on our existing knowledge to invent radical solutions, and we anticipate a need for artificial agents to do the same.

An important—but largely overlooked—challenge in computational creativity is cross-domain creativity, wherein a single agent or model is able to address creative problems from disparate domains. Commonly creativity researchers sidestep the need for general creative reasoning through hand-authoring of domain-specific knowledge. To the best of our knowledge this represents the first such cross-domain benchmark for computational creativity.

The rest of this paper is organized as follows: In section two we discuss related work and historic work that informs our position for this paper. In section three we discuss CrIB, all five problem categories and examples of each problem. In section four we demonstrate various baselines and features of the benchmark. We end with a discussion of the limitations of the benchmark, applications, and future directions.

[1] https://github.com/mguzdial3/CrIB

Related Work

Creativity Tests

There exist prior formal tests that involve computational models of creativity. For example the Lovelace 1.0 (Bringsjord, Bello, and Ferrucci 2003), Lovelace 2.0 (Riedl 2014) and MacGyver tests (Sarathy and Scheutz 2017) formalize bounds and loose evaluations that require creative cognition. However none of these prior approaches present sets of individual problems. Ravens Progressive Matrices (Raven and others 1938) has been used as a test for general cognitive ability, which includes creativity, most notably in work such as (Shegheva and Goel 2018). However, this test does not specifically seek to test creativity and only makes use of a single domain, whereas CrIB focuses on cross-domain, combinational p-creativity.

Combinational Creativity

There exists a range of combinational creativity techniques, which we briefly summarize. Notably researchers of combinational creativity do not frequently self-identify as addressing the same problem or field. Thus many combinational creativity approaches remain dependent on particular problem domains. However there has been some recent work to attempt to tie this field together (Guzdial and Riedl 2018).

Case-based reasoning (CBR) represents a general AI problem solving approach that relies on the storage, retrieval, and adaption of existing solutions (De Mantaras et al. 2005). The adaption function has lead to a large class of combinational creativity approaches, falling in two categories of either substitutional or structural adaption (Wilke and Bergmann 1998; Fox and Clarke 2009). These techniques tend to be domain-dependent, for example for the problem of text generation or tool creation (Hervás and Gervás 2006; Sizov, Öztürk, and Aamodt 2015).

Genetic Algorithms (GAs) represents a general AI problem solving approach that relies on an abstracted model of biological evolution (Srinivas and Patnaik 1994). It has proven extremely popular among computational creativity practioners, and we make use of it for an initial agent for solving CrIB. While not often recognized as such, the crossover function of a GA can be understood as a combinational creativity approach (Herrera, Lozano, and Sánchez 2003), though as with CBR adaption crossover functions tend to be domain-dependent.

Beyond CBR and GAs the area of belief revision, modeling how beliefs change, includes a function to merge existing beliefs with new beliefs(Konieczny, Lang, and Marquis 2004; Steels and De Beule 2006; Cojan and Lieber 2008; 2009; Konieczny and Pérez 2011). The mathematical notion of convolution has also been applied to blend weights, but with inconclusive results (Thagard and Stewart 2011).

We identify three combinational creativity approaches for further discussion given their popularity and generality across multiple problem domains. We visualize these approaches with illustrative examples in Figure 1.

Concept Blending Fauconnier and Turner (1998) formalized the "four space" theory of concept blending. They described four spaces: two *input spaces* represent the unblended elements, input space points are projected into a common *generic space* to identify equivalence, and these equivalent points are projected into a *blend space*. In the blend space, novel structure and patterns arise from the projection of equivalent points. Fauconnier and Turner (Fauconnier and Turner 1998; 2002) argued this was a ubiquitous process, occurring in discourse, problem solving, and general meaning making.

Concept blending typically requires a large amount of human authoring for individual concept spaces. More recent work has looked into automatically learning or deriving concepts (O'Donoghue et al. 2015; Guzdial and Riedl 2016). There has been work in blending individual tagged exemplars together based on surface level features of components (Alhashim et al. 2014). Fauconnier and Turner originally developed a set of heuristics for domain-independent measures of quality for blends, while more recent work has looked to introduce goals for blends (Li et al. 2012).

Amalgamation Ontañón and Plaza designed amalgams as a formal unification function between multiple cases (Ontañón and Plaza 2010). Similar to concept blending, amalgamation requires a knowledge base that specifies when two components of a case share a general form, for example "French" and "German" both share the more general form "nationality". Unlike concept blending, this shared generalization does not lead to a merging of components, but requires that only one of components be present in a final amalgam. For example, a "red French car" and an "old German car" could lead to an "old red French car" or an "old red German car".

Amalgams have been utilized as the adaption function in CBR systems (Manzano, Ontanón, and Plaza 2011), combined with concept blending for product development (Besold and Plaza 2015), and adapted to an asymmetrical form for story generation (Ontanón, Zhu, and Plaza 2012). Amalgamation represents a strong general method for combinational creativity. However it suffers from the drawbacks of other methods in terms of a traditional reliance on authored knowledge bases and domain-specific generalization.

Compositional Adaption Compositional adaption arose as a CBR adaption approach (Holland 1989; Fox and Clarke 2009), but has found significant applications in adaptive software (McKinley et al. 2004; Eisenbach, Sadler, and Wong 2007). The intuition behind compositional adaption is that individual concept components can be broken apart and recombined based on their connections. In adaptive software this process takes sets of functions with given inputs and outputs, and strings them together to achieve various effects, which makes compositional adaption similar to planning given a goal state or output. However, it can also be applied in a goal-less way to generate valid compositions.

Compositional adaption has been applied to recipe generation (Müller and Bergmann 2014; Badie and Mahmoudi 2017), intelligent tutoring systems (Reyhani, Badie, and Kharrat 2003), and traditional CBR approaches (Chedrawy and Abidi 2006). Unlike other methods compositional adaption does not require an explicit generalization knowledge

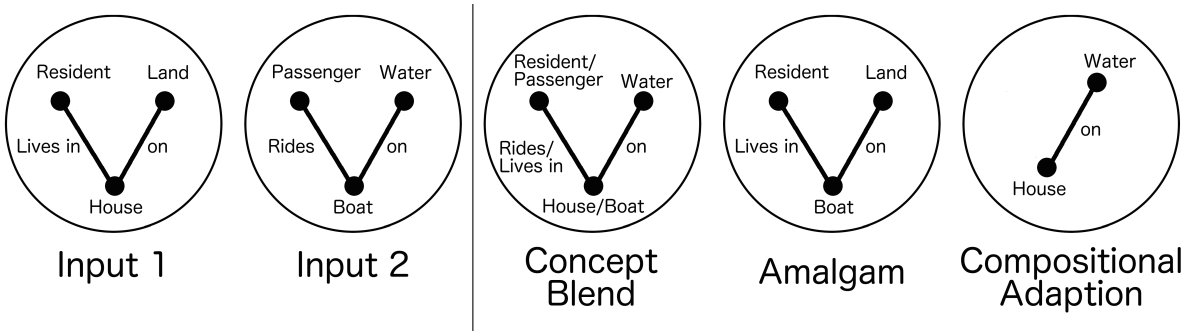

Figure 1: Example of three combinational creativity techniques. Two input spaces on left with example output from the three techniques on the right.

base. However, it is common to make use of a knowledge base to generalize across components and their relationships in order to expand the set of valid combinations.

Creative Invention Benchmark (CrIB)

In this section we discuss in more detail the Creative Invention Benchmark (CrIB). Our goal for the benchmark is to evaluate goal-driven combinational p-creativity, meaning a creative problem solving technique that relies on recombining the knowledge available to an individual. We refer to this class of problems as invention problems. To address our goal of generality we test combinational p-creativity across five distinct domains. This further reflects the multidisciplinary field of computational creativity. The domains are:

1. **Painting**, as in the running example in the introduction, in which an agent must invent new colors from some initial knowledge base to approximate some unknown goal painting.
2. **Alien language**, in which an agent must invent novel words to recreate an unknown goal sentence.
3. **Photobashing**, a practice from the field of concept art in which existing images are pieced together to form novel art. In this problem domain the agent must combine input images to approximate some unknown goal photobash.
4. **Narrative**, in which an agent, given a graphical representation of at least two story domains, must tell a target unknown goal story in some novel domain.
5. **Dessert Recipes**, in which an agent must combine existing recipe ingredients to create an unknown goal recipe.

The benchmark has a total of 2000 problems evenly spread across the five domains for a total of 400 problems per domain. For each problem an agent receives an initial knowledge base, a function to apply the agent's knowledge base in a domain-appropriate way (e.g. adding words to a sentence in the alien language domain), a function to clear the current state of the agent's submission in a domain-appropriate way (e.g. resetting the current canvas to a blank canvas in the painting domain), and a scoring function that measures the agent's distance to some unknown goal (with values ranging from 0.0 to 1.0).

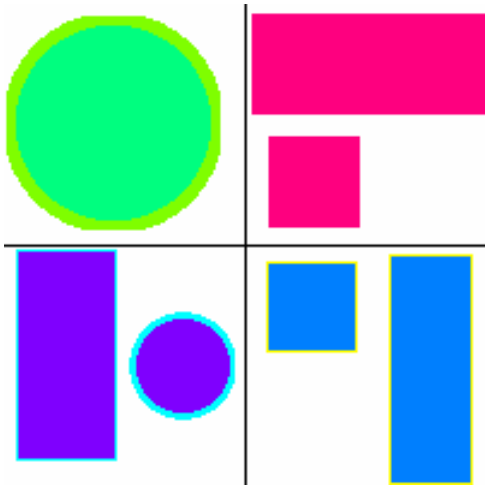

Figure 2: Four examples of unseen goal "paintings"

In the following subsections we discuss each domain in further detail. Notably we discuss the structure of each problem in terms of the input knowledge base, functions available to the agent, and the problem goal. We also discuss the approach taken to generate the domain problems and demonstrate an example problem. We note that all relevant code can be found at the public CrIB GitHub.

Painting

We include painting as a domain due to the long history of visual art in computational creativity, such as AARON (Cohen 1995) and the Painting Fool (Colton 2012). The painting problems of CrIB reflect the general problem description outlined in the introduction.

- **Input:** 2-6 colors as an initial knowledge base or palette.
- **Goal:** A painting that includes colors not included in the agent's initial knowledge base. The agent cannot directly access this goal painting.
- **Domain-Specific Function:** The agent is given a function **Paint** that takes as arguments two variables x and y (ranging from 0 to 1) that determine the location of a pixel and

Figure 3: Example of a human photobash from one of our artists on the right with the input images used on the left.

Figure 4: Example randomly selected generated photobash on the right, with the input images on the left.

a color (represented in RGB format). This function then sets that pixel to the specified color on an initially blank canvas of fixed size.

- **Clear:** This function allows the agent to clear the current canvas, resetting it to an all-white image.
- **Score:** The scoring function compares the current canvas with the target painting running vector subtraction for each pixel, it then sums over these values and normalizes to a maximum of 1.0. Without an agent inventing new colors, it is impossible to score a perfect 1.0 on any of these problems. However, it is possible to get to a relatively large local maxima.

We present examples of target images in Figure 2. To generate these problems we wrote a simple combination process that takes the primary (red, green, blue), secondary, and tertiary colors of a color wheel and finds all possible additive and subtractive combinations of colors (e.g. red and blue to make purple) From there it selects either a single color combination or multiple color combinations to draw upon for each problem, creating random geometric shapes with the target colors. We sorted the final 400 questions in terms of the number of initial colors and the number of colors and shapes in the target image as a stand-in for difficulty.

Alien Language

We include a fictional alien language as one of our domains as a stand-in for many language domains in the field of computational creativity such as human and musical language. In addition, making use of an alien language allowed us to include one problem domain in which the answers would be less obvious to a human, given that the language would follow artificial rules without a basis in real language. This allows us to consider a human subject study as a future baseline.

- **Input:** A set of 3-9 words as an initial knowledge base or vocabulary.
- **Goal:** A sentence that includes words not in the initial vocabulary, representenced as a sequence of words. This sentence is not directly accessible to the agent.
- **Domain-Specific Function:** The agent is given a function **AddWord** that takes as arguments one word from the knowledge base and adds it to the current sentence.
- **Clear:** This function allows the agent to clear the current sentence, resetting it to an empty sequence.
- **Score:** The scoring function compares the current sentence with the target sentence, giving a score of 1.0 for a perfect match, a 0.0 for no match, and a proportional score for partial matches of words in order in the sentences.

The alien language problems were generated by first generating a 2000-word vocabulary composed of randomly composing words from the characters ‘A’, ‘B’, ‘C’, ‘D’, ‘W’, ‘X’, ‘Y’, and ‘Z’ varying in length between two and twelve characters. From there we made use of arbitrary rules to compose a total of 400 target sentences varying in length between two and five words. For each target sentence we found one or two words in the sentence that could be considered combinations of between two and three other words in the vocabulary. For example “WAZZ” could be broken into ‘WA’ and ‘ZZ’. We then sorted each problem according to the number of input words and its length as a stand-in for difficulty. For example a simple sentence might be “WAZZ BYXBYW XDWB” with the initial knowledge base ‘BYXBYW’, ‘XDWB’, ‘WA’, and ‘ZZ’.

Photobashing

Photobashing is the practice of combining sets of input images to create a new composite image. It is a common practice for concept and key art for films, television shows, and video games. We include photobashing as it fits our general problem format and represents a real-world application of combinational creativity.

- **Input:** A set of 2-9 images as an initial knowledge base.
- **Goal:** A goal image that represents a combination of the input images. This image or photobash is not directly accessible to the agent.

- **Domain-Specific Function:** The agent is given a function **Stamp**, which takes as arguments x and y variables (ranging from 0.0 to 1.0) and one of the images of the knowledge base and places this image at an x,y location of an initially blank canvas of fixed size. Note that the agent can only add entire images from its knowledge base, meaning invention must occur to reach the goal.
- **Clear:** This function allows the agent to clear the current canvas, resetting it to a blank canvas.
- **Score:** The same as the painting scoring function.

To start generating photobashes we first gathered a palette of over eighty royalty-free stock images and photographs. We then made use of two distinct approaches to combine these images. For one we asked five human artists of a range of skill to construct photobashes. This lead to a total of 80 photobashes with a median value of 14 photobashes contributed across the five artists. An example of a human photbash can be found in Figure 3. For the remaining 320 photobashes we constructed a simple visual grammar by breaking apart a number of images of animals into heads, torsos, front legs and back legs. We then ran a script to combine these components. We required a human to verify the coherency of each generated photobash to ensure a baseline of quality. We reran the generation process for each rejected photobash. An example of a generated photobash can be found in Figure 4. The problems were sorted according to number of input images used to construct the goal as a stand-in for difficulty.

Narrative

We include narrative as a problem domain as it represents a common area of creativity research and allows us to include a novel representation. We made use of a story or plot graph representation as it encodes the branching nature of stories (Weyhrauch 1997). Plot graphs can be understood as a directed graph with the nodes as story events and the edges representing preconditions for events. Plot graphs represent multiple possible stories in a given domain and can generate stories by walking the graph.

- **Input:** A set of 2-4 distinct plot graphs
- **Goal:** A goal story represented as a sequence of events that cannot be generated from any of the input plot graphs. This story is not directly accessible to the agent.
- **Domain-Specific Function:** The agent is given a function **Submit**, which takes a single plot graph argument, and finds the closest story in the graph to the goal story. This closest story is set as the current story.
- **Clear:** This function removes any current story.
- **Score:** This function compares the current story and the target story. It returns 1.0 if the two match exactly, 0.0 if the two completely differ, and otherwise a proportional score for the number of shared events in sequence.

To begin the generation of narrative problems we first encoded ten existing published plot graphs in a common representation. We did this to ensure we did not accidentally encode too much stylistic similarity in the plot graphs. We pulled the movie, robbery, and pharmacy plot graphs from (Li 2015), the cat lover, cattle driver, stage coach and tour bus plot graphs from (Permar and Magerko 2013), the inheritance plot graph from (Min et al. 2008), the fantasy plot graph from (McIntyre and Lapata 2010), and the horror Anchorhead plot graph from (Nelson and Mateas 2005). For each plot graph we replaced the names of characters, each only had up to two, with 'A' and 'B'. We also simplified a few of the plot graphs such that each were at most 20 nodes. We then made use of amalgamation (Ontañón and Plaza 2010) to generate new plot graphs. To allow for mapping across different plot graphs we hand tagged certain event nodes with a higher-order theme (e.g. 'intro', 'ending', etc), additionally allowing mapping on shared words across nodes. From these plot graph amalgams we generated stories, which we then hand-checked to ensure coherency. For example a combination of fantasy and tourbus might output: "Monster holds B captive. A slays monster. A rescues B. A departs with B. A and B get married. A and B visit a Landmark." We sorted these problems according to the number of initial plot graphs used to create the goal story's plot graph.

Dessert Recipe

For our final domain we chose recipes, more specifically dessert recipes, as recipes represent a common example domain for adaption and creativity. This also allowed for a second real-world domain beyond photobashing. For each dessert recipe problem the agent must invent a recipe given existing recipes.

- **Input:** A set of 3-130 distinct recipes encoded as a recipe name and a set of ingredients (e.g. banana muffins (bananas, flour, eggs, milk, sugar)).
- **Goal:** A goal recipe distinct from all of the input recipes. This goal recipe is not directly accessible to the agent.
- **Domain-Specific Function:** The agent is given a function **Submit**, which takes a single recipe argument. This is set as the current recipe.
- **Clear:** This function removes any current recipe.
- **Score:** This function compares the current and target recipe ingredients. It returns a value between 0 and 1 dependent on the extent to which the two sets overlap.

To generate these problems we drew on the dessert dataset from (Veale 2017). For each dessert we found all sets of other desserts whose ingredients could be composed to match its ingredients. From this point it was simple to randomly select a set of four hundred of these possible compositions for each problem. We then sorted these problems according to the number of initial desserts in the knowledge base as a stand-in for difficulty. This number varied massively from 3 to 142. As an example given banana muffins (bananas, flour, eggs, milk, sugar), Vanilla wafer cake (shredded coconut, flour, milk, eggs, sugar, chopped pecans, vanilla essence), and treacle tart (golden syrup, lemon zest, butter, flour) produce pound cake (butter, sugar, eggs, flour, vanilla essence).

Table 1: Average output of two baselines and the random agent for each domain and across all five domains.

	Painting	Language	Photobash	Narrative	Dessert	Total
Null	0.70	0.0	0.76	0.0	0.0	0.29
Uncreative Max	0.85	0.72	0.89	0.45	0.49	0.61

Table 2: Scores for the presented agents and their average total.

	Painting	Language	Photobash	Narrative	Dessert	Total
Random Agent	-0.99	-2.42	-1.50	-0.32	-0.50	-1.15
GA_{100}	-0.99	-1.41	0.02	0.76	0.35	-0.25
GA_{1000}	-0.91	-1.19	0.17	0.81	0.35	-0.14

Using CrIB

In this section we discuss how to make use of CriB. We introduce two baselines to better characterize the benchmark, introduce a scoring function that relies on one of these two baselines, and present two initial agents that attempt to solve the benchmark.

Baselines

In this section we demonstrate two baselines to further characterize CrIB. The baseline "null" represents the score of an agent that does absolutely nothing. The baseline "Uncreative Max" represents the best an agent could do without any invention of additional knowledge beyond the initial input knowledge base for each problem. We constructed Uncreative Max by finding the closest element of the initial knowledge base to the target concepts.

We summarize the average scores of these two baselines in Table 1. We note that the two visual domains—painting and photobashing—can achieve the highest values since they only look at pixel-by-pixel comparisons and share many white pixels. In addition, it is relatively easy to score high on the alien language domain since the goal sentences are composed mostly of words from the initial knowledge base. However, narrative and dessert generation are far less successful. Our baselines are not meant to signify any intelligence, but to provide a means for analyzing how easy it is to guess a high-scoring solution without creative reasoning if we attempt to score naively.

Scoring Function

The prior section demonstrates that it is possible to get high scores without creative behavior if we score naively. However, we intend this benchmark to measure a facet of creativity. Therefore we use the following scoring function for each problem domain:

$$Score = (NScore_a - NScore_u)/(400 - NScore_u)$$

Where $Score$ represents our final score, $NScore_a$ represents the naive score discussed for each domain above for some current agent a, $NScore_u$ represents the naive score discussed above for the Uncertain Max baseline. In other words an agent's actual score is the amount that it does better than Uncreative Max. We are essentially making the assumption that if the score of Uncreative Max represents uncreative computation, whatever is left must require creative computation. Because Uncreative Max makes use of all available knowledge without any invention of new knowledge, an agent may receive a negative score if it fails to make use of all of the knowledge it is initially given.

Initial Agents

We present two initial agents as a means of demonstrating that the problems of this benchmark are non-trivial. For the first agent we present a random agent that randomly selects a single element of the initial knowledge base and runs the domain-specific function. We note that this first agent cannot be expected to do better than Uncreative Max, but we include it in order to compare it to our second agent. Our second agent is a genetic algorithm (GA) agent, which we tested in two variations.

The GA agent searches in the space of possibile final answers relevant to each domain (images for painting and photobashing, sentences for alien language, recipes for dessert recipe, and stories for narrative). It uses a mutation function that randomly swaps out some value of the current representation with a value from the knowledge base. It uses a crossover function that randomly selects values from two parents, selected according to current naive score, to fill in the variables of a new child representation (e.g. randomly grabbing words from two parent sentences to create a child sentence). We used a mutation rate of 0.7, and selected the 20 best parents to create 20 new children with each iteration. We created two variations on this agent based upon number of iterations and population size. For the first GA_{100} we ran the GA for a maximum of 100 iterations with a population of 100 individuals. For the second GA_{1000} we ran for a maximum of 1000 iterations with a population of 1000 individuals. We present the scores of all agents in Table 2.

We note a number of interesting results comparing the scores across these agents. GA_{1000} did the best, as one might expect, but did far worse than one might naively assume. The primary reason for this was that the simple mechanism by which both GA agents introduced new knowledge (random mutations and crossover) was insufficient to produce the desired combinations given the feedback of the scoring function. This is most clear in comparing GA_{1000} and GA_{100} in terms of the Dessert Recipes and Narrative performance. In the former there was no improvement in the score despite a tenfold increase in iterations and population.

The most successful domain was Narrative, since the agent's crossover and mutation functions were well-suited to swapping out events in a story. We found with additional tests that the GA_{1000} values largely represent the upper-bound of this approach, indicating that solving this benchmark is not simply a problem of longer training time.

Ways of Using CrIB

We include all of the discussed agents and baselines and a few additional agents on the public GitHub. Beyond reporting scores we recommend researchers make use of these given agents to draw comparisons. In particular beyond score we recommend reporting the average increase in size of the knowledge base per problem and the number of guesses or training steps necessary to achieve the reported scores. These features can allow for better comparison in terms of an agent's ability to make insightful or human-like combinations quickly. In terms of formats for reporting results we anticipate that this will depend on the agent. One clear approach would be to make use of Reinforcement Learning, which might involve reporting average score over time. Alternatively one might approach this problem with a more traditional classifier, at which point reporting training and testing error may be appropriate.

We note that one naive approach might be to hand-author knowledge for each domain. For example, simply giving an agent all primary, secondary, and tertiary colors for the painting domain. However, this goes against the spirit of the benchmark, and entirely removes any need for creative reflection or invention from an agent.

Limitations and Future Work

We note that the benchmark at present has a number of limitations. We do not present any successful, creative agents by our own measures in this paper. The development of such agents remains the largest area of future work Further, while relatively large at first glance 2000 problems is small compared to similar benchmarks in other domains. Notably it would be trivial to expand the painting, alien language, and the dessert recipe domains to many times their current size, which one can accomplish given the GitHub generator code. However the need for human evaluation for narrative and photobashing represents a limiting factor.

There are many more possible domains we could include in this benchmark. For example music and product generation, both common computational creativity domains. We fully intend to expand CrIB in future versions.

Conclusions

We present the Creative Invention Benchmark (CrIB), a benchmark for evaluating combinational p-creativity. We demonstrate the generative process for creating the 400 problems for each of the five domains of the benchmark, and the performance of a set of baselines and agents. We make this baseline available to the general research community through GitHub, and hope that it inspires further developments in the field of computational creativity.

Acknowledgments

We gratefully acknowledge the NSF for supporting this research under NSF award 1525967. We appreciate the detailed work of the ICCC reviewers, whose insight greatly improved this final paper. In addition we would like to especially thank Jack Yardley Ingram, without whom this paper would not have been finished.

References

Alhashim, I.; Li, H.; Xu, K.; Cao, J.; Ma, R.; and Zhang, H. 2014. Topology-varying 3d shape creation via structural blending. *ACM Transactions on Graphics (TOG)* 33(4):158.

Badie, K., and Mahmoudi, M. T. 2017. Compositional adaptation in case-based reasoning based on the semantic relations between the components in the cases. In *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 449–454. IEEE.

Besold, T. R., and Plaza, E. 2015. Generalize and blend: Concept blending based on generalization, analogy, and amalgams. In *ICCC*, 150–157.

Boden, M. A. 1998. Creativity and artificial intelligence. *Artificial Intelligence* 103(1-2):347–356.

Boden, M. A. 2004. *The creative mind: Myths and mechanisms*. Psychology Press.

Bringsjord, S.; Bello, P.; and Ferrucci, D. 2003. Creativity, the turing test, and the (better) lovelace test. In *The Turing Test*. Springer. 215–239.

Chedrawy, Z., and Abidi, S. R. 2006. Case based reasoning for information personalization: using a context-sensitive compositional case adaptation approach. In *Engineering of Intelligent Systems, 2006 IEEE International Conference on*, 1–6. IEEE.

Cohen, H. 1995. The further exploits of aaron, painter. *Stanford Humanities Review* 4(2):141–158.

Cojan, J., and Lieber, J. 2008. Conservative adaptation in metric spaces. In *European Conference on Case-Based Reasoning*, 135–149. Springer.

Cojan, J., and Lieber, J. 2009. Belief merging-based case combination. In *ICCBR*, 105–119. Springer.

Colton, S. 2012. The painting fool: Stories from building an automated painter. In *Computers and creativity*. Springer. 3–38.

De Mantaras, R. L.; McSherry, D.; Bridge, D.; Leake, D.; Smyth, B.; Craw, S.; Faltings, B.; Maher, M. L.; T COX, M.; Forbus, K.; et al. 2005. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review* 20(3):215–240.

Eisenbach, S.; Sadler, C.; and Wong, D. 2007. Component adaptation in contemporary execution environments. In *Distributed Applications and Interoperable Systems*, 90–103. Springer.

Fauconnier, G., and Turner, M. 1998. Conceptual integration networks. *Cognitive science* 22(2):133–187.

Fauconnier, G., and Turner, M. 2002. *The way we think: Conceptual blending and the mind's hidden complexities*. Basic Books.

Fox, J., and Clarke, S. 2009. Exploring approaches to dynamic adaptation. In *Proceedings of the 3rd International DiscCoTec Workshop on Middleware-Application Interaction*, 19–24. ACM.

Guzdial, M., and Riedl, M. 2016. Learning to blend computer game levels. In *Proceedings of the Seventh International Conference on Computational Creativity (ICCC 2016)*.

Guzdial, M., and Riedl, M. O. 2018. Combinatorial meta search. *Proceedings of the 1st Knowledge Extraction from Games Workshop*.

Herrera, F.; Lozano, M.; and Sánchez, A. M. 2003. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems* 18(3):309–338.

Hervás, R., and Gervás, P. 2006. Case-based reasoning for knowledge-intensive template selection during text generation. In *European Conference on Case-Based Reasoning*, 151–165. Springer.

Holland, J. H. 1989. *Induction: Processes of inference, learning, and discovery*. Mit Press.

Konieczny, S., and Pérez, R. P. 2011. Logic based merging. *Journal of Philosophical Logic* 40(2):239–270.

Konieczny, S.; Lang, J.; and Marquis, P. 2004. Da2 merging operators. *Artificial Intelligence* 157(1-2):49–79.

LeCun, Y. 1998. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Li, B.; Zook, A.; Davis, N.; and Riedl, M. O. 2012. Goal-driven conceptual blending: A computational approach for creativity. In *Proceedings of the 2012 International Conference on Computational Creativity, Dublin, Ireland*, 3–16.

Li, B. 2015. *Learning knowledge to support domain-independent narrative intelligence*. Ph.D. Dissertation, Georgia Institute of Technology.

Manzano, S.; Ontanón, S.; and Plaza, E. 2011. Amalgam-based reuse for multiagent case-based reasoning. In *International Conference on Case-Based Reasoning*, 122–136. Springer.

McIntyre, N., and Lapata, M. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 1562–1572. Association for Computational Linguistics.

McKinley, P. K.; Sadjadi, S. M.; Kasten, E. P.; and Cheng, B. H. 2004. A taxonomy of compositional adaptation. *Rapport Technique numéroMSU-CSE-04-17*.

Min, W.-H.; Shim, E.-S.; Kim, Y.-J.; and Cheong, Y.-G. 2008. Planning-integrated story graph for interactive narratives. In *Proceedings of the 2nd ACM international workshop on Story representation, mechanism and context*, 27–32. ACM.

Müller, G., and Bergmann, R. 2014. Compositional adaptation of cooking recipes using workflow streams. In *Computer cooking contest, workshop proceedings ICCBR*.

Nelson, M. J., and Mateas, M. 2005. Search-based drama management in the interactive fiction anchorhead. In *AIIDE*, 99–104.

O'Donoghue, D. P.; Abgaz, Y.; Hurley, D.; and Ronzano, F. 2015. Stimulating and simulating creativity with dr inventor. In *Proceedings of the 6th ICCC*.

Ontañón, S., and Plaza, E. 2010. Amalgams: A formal approach for combining multiple case solutions. In *Case-Based Reasoning. Research and Development*. Springer. 257–271.

Ontanón, S.; Zhu, J.; and Plaza, E. 2012. Case-based story generation through story amalgamation. In *Proceedings of the ICCBR 2012 Workshops*, 223–232.

Permar, J., and Magerko, B. 2013. A conceptual blending approach to the generation of cognitive scripts for interactive narrative. In *Proceedings of the 9th AIIDE Conference*.

Raven, J. C., et al. 1938. *Raven's progressive matrices*. Western Psychological Services.

Reyhani, N.; Badie, K.; and Kharrat, M. 2003. A new approach to compositional adaptation based on optimizing the global distance function and its application in an intelligent tutoring system. In *Information Reuse and Integration, 2003. IRI 2003. IEEE International Conference on*, 285–290. IEEE.

Riedl, M. O. 2014. The lovelace 2.0 test of artificial creativity and intelligence. *arXiv preprint arXiv:1410.6142*.

Sarathy, V., and Scheutz, M. 2017. The macgyver test-a framework for evaluating machine resourcefulness and creative problem solving. *arXiv preprint arXiv:1704.08350*.

Shegheva, S., and Goel, A. 2018. The structural affinity method for solving the raven?s progressive matrices test for intelligence. In *Proceedings of Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Sizov, G.; Öztürk, P.; and Aamodt, A. 2015. Evidence-driven retrieval in textual cbr: bridging the gap between retrieval and reuse. In *International Conference on Case-Based Reasoning*, 351–365. Springer.

Srinivas, M., and Patnaik, L. M. 1994. Genetic algorithms: A survey. *computer* 27(6):17–26.

Steels, L., and De Beule, J. 2006. Unify and merge in fluid construction grammar. *EELC* 4211:197–223.

Thagard, P., and Stewart, T. C. 2011. The aha! experience: Creativity through emergent binding in neural networks. *Cognitive science* 35(1):1–33.

Veale, T. 2017. Déjà vu all over again.

Weyhrauch, P. 1997. *Guiding Interactive Fiction*. Ph.D. Dissertation, Ph. D. Dissertation, Carnegie Mellon University.

Wilke, W., and Bergmann, R. 1998. Techniques and knowledge used for adaptation during case-based problem solving. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 497–506. Springer.