Computer Creativity in Games—How Much Knowledge Is Needed?

David C. Moffat and Paul Hanson

Abstract In seeking to make computer systems that can be creative, there are several problems, including the definition of the concept of creativity, understanding how to design creative algorithms, encoding the domain knowledge needed for them, and evaluating the creative products of those algorithms. We make a case that games are in some ways a good domain in which to research and develop creative algorithms, in particular the design of computer programs that can play games in ways that human players would consider to be creative. Although the playing of games is not in itself a very important research aim, we argue that it is a good arena in which to test out attempts to make creative programs. We initially consider how the field of computer play may be said to have contributed to computational creativity so far, with games like chess that require a specialised knowledge base in order to perform well. As an example, we show the play of our program for a falling-blocks puzzle game, that uses simple search to play to a fairly good standard, and which a human player may well consider to be creative at times. In that case, we conclude that at least in some non-trivial domains, computational creativity might be achieved without the typically heavy requirement for a large or sophisticated knowledge-base or even machine learning.

1 Creativity—Concept and Requirements

1.1 Definition of Creativity

In the literature for creativity (and computational creativity), if a product or idea is generally said to be the result of a creative thought process then it must be both *novel*, and *valuable* (e.g. [1, 5]). What counts as valuable will of course depend on the domain; and the novelty of any idea should somehow be calculated by comparison with previous ideas in the domain. In many or most domains these are vague,

D.C. Moffat (✉) · P. Hanson
Department of Computing, Glasgow Caledonian University, Glasgow, UK
e-mail: D.C.Moffat@gcu.ac.uk

S. Kunifuji et al. (eds.), *Knowledge, Information and Creativity Support Systems*,
Advances in Intelligent Systems and Computing 416,
DOI 10.1007/978-3-319-27478-2_30

informal and dependent on the intuition of domain experts. For example, this is the case for all the arts, clearly; and for practically all the so-called "creative industries." Indeed, we always start with human intuition as our only guide to what is creative, until we can develop more formal measures.

Some authors include further conditions alongside novelty and value. Notably, Boden says that a creative idea should generally be in some sense *surprising* [2]. This could mean that the idea is not obvious even to domain experts, and that they would not easily have thought of it themselves. It could also mean that the creative idea might seem elegant, once it is heard. In the more striking cases, it might then even seem obvious; but that is only with the benefit of hindsight.

While there remains debate about the definition of creativity, we shall take it to mean those three conditions, of *novelty*, *value* and *surprise*, with *elegance* an optional fourth. As well as the matter of definition, in the field of computational creativity (CC), there is the difficult matter of designing *algorithms* that could be capable of showing something like a human level of creativity; and the vexatious need to *evaluate* the creative products, even without any clear criteria that can be formulated.

1.2 Requirement for KA (Knowledge Acquisition)

As well as designing algorithms for CC, a prior problem is to provide them with any necessary knowledge to work with. When we aim to make programs that independently contribute to a field of human endeavour, they also need to have all or most of the knowledge that humans already discovered, in that field. This is a formidable requirement, in the general case requiring us to formalise a suitable knowledge representation scheme and then encode in it all relevant human knowledge.

It is essentially the AI version of the philosophical knowledge problem, whose best known and most ambitious work was the *Cyc* project, started in 1984, which has taken an open-source form as *OpenCyc*. The difficulties such projects experience in creating their large knowledgebases, and trying to ensure that they do not permit dubious inferences to be drawn, are a reflection of the inherent difficulty of knowledge. It was a surprise to AI researchers, for example, that common-sense reasoning turned out to be intractable so far, on realistic domain sizes.

This venerable old problem of AI and expert systems is still with us today in most domains of real-world expertise. Even if we had arrived at a fully competent knowledge representation language for all domains, the task of encoding any single domain into it would be extremely intimidating. It is often referred to as the "knowledge acquisition (KA) bottleneck."

Instead of encoding knowledge into a system (or as well as doing that), it is possible to make the system learn for itself, from its own experience in the domain. This approach of machine learning, however, is no nearer a general solution than the more direct approach of encoding. There being as yet no general solution to the KA-bottleneck, we have to choose our domain for creativity carefully, so that we can feasibly solve the KA issue for the chosen domain at least.

1.3 Requirement for Evaluation

Another reason to choose the creative domain carefully is to enable us to evaluate the ideas or other creative products as reliably and as uncontroversially as possible. The evaluation should also be quick, and formalisable, especially in CC fields, to incorporate them into the creative generation algorithms.

In general, however, it is not easy to find a consensus on what ideas in any given domain are truly *novel*, *valuable*, and *surprising* (not to mention *elegant*). Nor can we usually agree on a suitable formalisation of these concepts, which are still an active research area.

Therefore we generally fall back on naive human judgement, possibly with several judges, at least to begin with, in order to conduct at least some kind of evaluation of creative programs. This is the approach taken in this paper.

2 Gameplay as a Domain for Computational Creativity

Historically, games like chess have been important in the development of AI, leading to today's sophisticated algorithms that can beat the strongest human players. The way that these algorithms work is by a highly specialised kind of search process, that searches the very widely branching game tree guided by heuristic evaluations of the anticipated board positions. There is a lot of human knowledge and gameplay intuitions encoded into these heuristic functions. There is also a large component of machine learning, as the algorithms can play against themselves at high speed to better tune their functions. Despite the heavy reliance of the best chess programs on both expert knowledge and machine learning, however, we consider that such a reliance is not actually necessary for games, and so we select a game that can be tractable without demanding these functions.

Here we intend for game playing programs to be creative, just in the way they play their games. There are other ways to apply creative programs to the domain of games, such as by helping to design them (so called "procedural content generation"); but that is not the focus of this paper.

The attractions of gameplay as a testbed for CC (computational creativity) research are that it offers at least partial solutions to most of the above problems.

Firstly, games are typically easy to formalise briefly and clearly, because they are designed to have simple and exact rules. As games are often highly abstracted from their original worldly metaphor (e.g. real armies at war, in the case of chess), they do not require mountains of common sense or other domain knowledge to be encoded. Many games thus avoid or alleviate the *KA-bottleneck problem*.

Secondly, the playing of games has been a testbed application of AI since the beginning, so we have a depth of knowledge about how to write algorithms for this task. This largely solves the *algorithm problem*.

Thirdly, each game situation (or position) is potentially very similar to many that have occurred before in other games; and yet the smallest difference might have big consequences, so that there is rich novelty available, even against any database of games played in the past. Given such a database, it should be feasible to program a variety of possible estimates of *novelty*. Games typically award points or have other ways to quantify play performance, which can conveniently be used to assess *value*. Taken together, these two criteria of creativity are thus fairly measurable, which goes a long way to solving the *evaluation problem*.

Finally, the criterion of *surprise* or even *elegance* are less easy to formalise. We shall therefore use the game score as a basis for value, and rely on human intuitive judgement for the criteria of novelty, surprise or elegance.

3 PuyoPuyo as a Game for CC Research

The subject of our study is a simple arcade game called PuyoPuyo. First we implement a competent player for it. We then illustrate its performance that could be said to be creative gameplay, and end with some of its potential advantages as a testbed for CC research.

3.1 The Game Puyo-Puyo

PuyoPuyo is played at the same time by two opposing players, where each player has a game board of 6×13 spaces. However, our program only plays a simpler, single-player version, so we here leave out the features of the game which are relevant when there are two players.

Pairs of different coloured puyo are dropped onto the player's board, which must then be positioned to avoid filling up the game board. As in *Tetris*, if and when the board fills up entirely to the top, the player loses [8]. Spaces on the board can be cleared by popping four or more puyo of the same colour. When this happens, the player's score increases. After a chain of Puyo is cleared in this way, any other puyo above them fall into their spaces (again, similar to *Tetris*); and if that makes a new chain (of four or more puyo of another colour), then it is also cleared away, and so on until no more puyo can be cleared. Then the next pair of puyo enter the top of the board for the player to drop.

In the example a blue and green puyo pair is dropped. The blue one completes a chain, which then disappears to let the higher puyos fall. A green chain is thereby completed, which also disappears. This causes a blue chain and then a red chain to go as well, leaving only five puyos at the end of the move. Because four chains have gone in a single move, it has got a high score (much more than the four chains would have got separately). To build up the potential for such "cascades" of chains is thus the strategic aim of the game (Fig. 1).

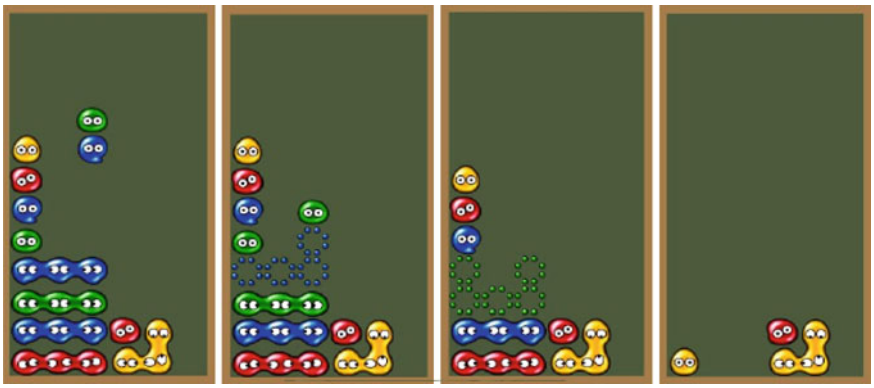

Fig. 1 Screenshots of *PuyoPuyo*. A *green/blue* pair falls, to make a chain of four *blues*, which clear away; then the five *greens* go; then the next four *blues*; and finally four *reds*

While the aim may be clear, it is unclear how to achieve it. In other games like *Tetris*, one can postulate general rules or heuristics about good ways to play the game. Players soon learn to fit blocks in patterns and wait or hope for blocks to fall that exactly fit into the gaps, so that they can clear lines. But in *PuyoPuyo* there is not such an obvious strategic method. We thus fall back on brute search methods, which are not guided by expert heuristics. It is for these reasons that MC algorithms are interesting to investigate.

3.2 Monte Carlo Search (MCS) for PuyoPuyo

Monte Carlo (MC) simulations are predominantly used as a substitute to heuristic evaluation functions in game AI—the advantage being that a value in a search tree can be determined without the need for complex, time consuming expert knowledge based functions. In order to capitalize on the advantages of MC simulations and to improve the search capabilities of it, the concept of MC evolved into what is known as the Monte Carlo Tree Search (MCTS).

The MCTS is a form of best-first search which uses the results of multiple MC simulations to determine which path to take [4] This form of search was used by Bouzy [3] who combined a standard MC search, with a shallow and selective global tree search. His ComputerGo programs Indigo and Olga play the game pretty well [3].

In this study we investigate the possible value of a simpler MC algorithm, as follows, which shows how even a small and simple algorithm can play with some apparent creativity.

Because the game *PuyoPuyo* has a large branching factor it becomes computationally difficult to exhaustively search the tree more than a few layers deep. Each

new pair of *puyo* can be rotated four ways (similar to *Tetris*), and dropped in any of the six columns. Less the two positions where one of the *puyo* would be off the edge of the board, this means that the top level of the search tree branches 22 ways (i.e. $4 \times 6 - 2$). In this study the simple MCS algorithm only searches the first layer exhaustively, by visiting each of the 22 children in turn, in round-robin fashion. None of the nodes are favoured either; in contrast to what happens with many MCTS algorithms, where more attractive looking nodes are visited more often under the assumption that the best solutions may be found under those ones. These two simplifications of the more complex and advanced MCTS family of algorithms make this MCS algorithm much shorter to implement in code.

Below the first layer, the MCS algorithm switches into random search mode in which it simulates further plays of the game, by visiting and expanding only one child at random in each layer. However, it is not possible in *PuyoPuyo* to complete each so-called *playout* to the end of the game, until a terminal node is reached with a win-lose result. This is because the game can only terminate in a loss for the player, whose aim is to survive as long as possible, just as in other falling-block games like *Tetris*; and that loss should be far into the future, and well beyond the depth that could guarantee to be reasonably searched in a fixed time limit.

In this implementation of MCS therefore, we impose a depth limit to the search and take the increment in score when the limit is reached as our indication of how good the partial playout (the simulated random play to that depth) was. This score is then used to update the estimated value of the initial move at the first layer that preceded the random playout stage. Each of the 22 first-level nodes (children of the root node) is annotated with its own "best so far" score, from all the playouts that have yet originated from it. In this way the algorithm may be interrupted at any time, and the best of all those first-level nodes can be selected as the best move to play. As the game is to be played in real-time, the MCS algorithm is allowed to run for as long as it can, until the blocks have fallen so far that the AI player must now commit itself to one of the 22 possibile moves, and play it.

After some experimentation, we found that a depth-limit of only three was enough to give a good performance with real-time play on a modest PC (with Intel's Celeron two-core processor). This happens to be the most authentic depth-limit, as it corresponds to being aware of the next two blocks that will come after the current one. As a human player of PuyoPuyo can see the next two blocks queueing up at the side of the board (similar to *Tetris* again), this means the AI player is playing under the same conditions as the human player. More detail about the algorithm itself can be found in our earlier paper [6].

4 Performance of the PuyoPuyo Player

The MCS algorithm described plays PuyoPuyo pretty well. An example of one of its more surprising and possibly creative moves, see Fig. 2.

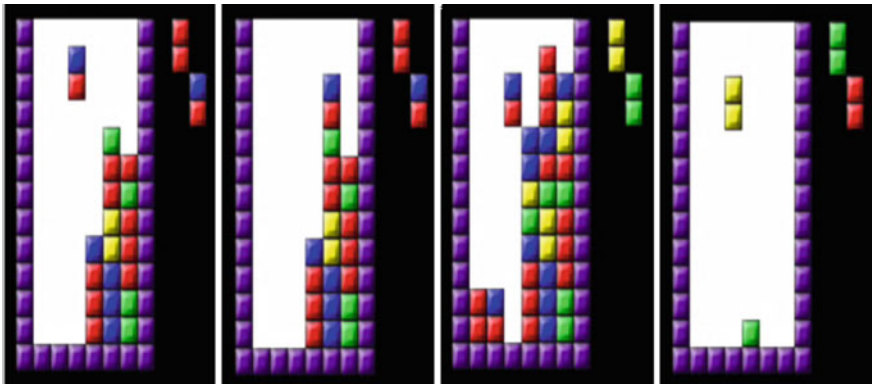

Fig. 2 Program playing *PuyoPuyo*. The program chooses a sequence of less obvious moves that seem to risk losing, but builds a good position on the *left* of the board, and finally exploits it to clear the whole board

In the figure we see that a good move would be to drop the red/blue pair from where it is, as that would clear first the reds, and then more chains after that, to get a good score. But the program instead moves the pair to the side, and then moves the next few pairs to the sides as well; but putting more blues and reds to the left. This endangers the play as the blocks nearly reach the top; but then the last pair shown (the blue/red) is dropped right into the hole, and nearly the entire board is cleared in one go.

This piece of play seems *elegant*, even to the two authors who are novice PuyoPuyo players; as well as apparently risky, which makes the final clearance more satisfying. As a high score is achieved by these moves, and a more obvious and still good move was foregone to make this play, it seems to be a good candidate for a creative play.

Although it may appear that the program is planning a long way ahead, it is in fact only searching to depth three. The reason that it appears to be conceiving of the game in terms of strategies like "building a strong position to the left of the board" and tactics like "aim to plug the gap with same coloured puyo" is probably due to our own human tendency to conceive the game in those terms. The program itself has no such complexities within it, however. It only has knowledge of the rules of the game, and it only looks three moves ahead. Anything else it appears to be doing must therefore be an artefact, or a result of the way we perceive the game.

5 Discussion

An earlier version of this paper was presented at the KICSS-14 conference [7], where it generated some interesting reactions that are worth reviewing here. This included both questions and discussion after the presentation, and comments by reviewers previously.

Questions were raised regarding the claim that little or no knowledge is required for creativity, at least in the restricted domain chosen of games, and in particular the game of *PuyoPuyo*. Some people wished that the amount of knowledge had been quantified more formally.

Because of the lack of *formal* evaluation, there was also some feeling by a reviewer that the case made by the paper was unconvincing, in that the creativity was not demonstrated. However, at the presentation there was no such doubt raised, perhaps because the video recording of the game as it was played was more convincing to the audience than merely reading about it as described in the paper.

In fact, the performance of the AI player seemed to impress all who saw it, just as we had hoped it might. After all, we ourselves as the authors found the play to be strikingly suggestive of creativity, and that it would probably have been deemed creative if played by a human. So it was good to see that the audience who saw the presentation at least seemed to agree thus far. But they began to doubt the creativity in the play once they knew how the AI player was made. One reaction was that the AI player was not properly creative as it had very little knowledge of the game it was playing, or how to play it. This reaction was gratifying to us, as it was in fact the main point of the paper. A related objection was that the AI player could not be truly creative because it was based on a blind search process. Again however, that was actually the point of the paper: to show that little knowledge was needed, and only a simple search strategy, to give the appearance of creative play.

Regarding the quantification of knowledge, we see no need for a formal treatment in this case. It is clear that the AI player has hardly any knowledge, since it is based on a Monte-Carlo search algorithm. It is the conflict between this lack of knowledge and the expectations based on theory (as well as the intuitions of the audience reactions), that was our purpose to establish.

Another reaction expressed, on hearing how the algorithm worked, was to assert that it could not be genuinely creative because it was determined in its behaviour, and would play exactly the same way in all future games, with identical initial circumstances. It is curious to note that this objection is nearly the opposite to the other objection raised above, which found the algorithm uncreative because it was too random in its progress.

However, these objections are all beside the point, it seems to us. The message of the paper is that the algorithm achieves its performance by means of a guided random procedure and no expert knowledge; and that the performance is judged to be creative simply enough by human intuition; first our own, and then the audience at the presentation. All objections that doubt this creative performance, only on the basis of knowledge about how the program is coded, are judging the matter on privileged information, which should strictly speaking be disregarded. If a program is to be thought uncreative simply because it is just a program, whether random or not, then no program will every be judged creative. The field of computational creativity would be doomed to fail from the outset in that case, so it cannot be a fair way to judge the matter.

Although it may seem counterintuitive to many that a program can be creative, and even more so that it can be so without any expert knowledge, and with only the help of a simple search algorithm; nevertheless that is the conclusion of the argument in this paper. The evidence is the AI player described and demonstrated, and its play that appears to be creative as judged by human observers.

6 Conclusion

Having identified some key ways in which KA problems are presenting some major road-blocks to CC research, we argued for the role of games (and in particular gameplay) as a suitable domain.

We presented a simple game, *PuyoPuyo*, that can be understood by anybody who knows *Tetris*, but that does not yet have strategies formulated for how to play it well. Therefore a correspondingly simple Monte-Carlo search algorithm was encoded to play it, which had no special game strategy or other heuristic knowledge put into it. The algorithm did not learn either; and yet it was able to achieve a creditable performance. The example play of the game showed that the program is capable of some apparently novel, strong and even surprising play. In other words its play appears to be creative to some extent.

The main significance of the work as we see it, is that for the chosen game and algorithm, a modestly creative performance was achieved without any special knowledge, and without any machine learning.

Furthermore, the level of game expertise required by the CC researchers, who work with such a domain, is low; as is the complexity of the gameplay algorithm. This should allow more focused attention on the key research issues of how best to assess and account for the roots of any creativity found, which could help the research field to progress faster.

Common reactions to this kind of work sometimes appear to be based on preconcieved notions about what creativity is and how it is achieved. However, they can be misleading and might lead to unfair judgements. The outputs of a creative program should be judged without reference to how they were achieved, or even as if they were produced by a human. In that case, the AI player shown here would be seen as creative, even though it was produced with little knowledge, and no learning. This goes against the prevailing views in the field, and is thus the contribution of this work.

References

1. Boden, M.A.: The creative mind. Abacus, London (1998)
2. Boden, M.A.: Creativity and artificial intelligence. Artif. Intell. **103**, 347–356 (1998)
3. Bouzy, B.: Associating shallow and global tree search with Monte Carlo for 9×9 go. In: International Conference on Computers and Games, LNCS 3846, pp. 67–80. Ramat-Gan, Israel (2004)

4. Chaslot, G.M.J.B., Winands, M.H.M., van den Herik, H.J., Uiterwijk, J.W.H.M., Bouzy, B.: Progressive strategies for Monte-Carlo tree search. New Math. Nat. Comput. **4**(3), 343–357 (2008)
5. Csikszentmihalyi, M.: Creativity: Flow and the Psychology of Discovery and Invention. Harper-Perennial, New York (1997)
6. Hanson, P., Moffat, D.C.: Monte Carlo search for a real-time arcade game, puyo-puyo. In: Proceedings of the Symposium on AI Games. AISB 2014 Convention, at Goldsmiths, University of London (2014)
7. Moffat, D.C., Hanson, P.: Computer creativity in games—how much knowledge is needed? In: Papadopoulos, G.A. (ed.) Proceedings of the 9th International Conference on Knowledge, Information and Creativity Support Systems, pp. 346–351. Limassol, Cyprus, 6–8 Nov (2014). http://kicss2014.cs.ucy.ac.cy
8. Sonic Team: PuyoPuyo. Sega (2009). Cartridge; Nintendo DS