Minstrel Remixed: Procedurally Generating Stories

Brandon Tearse, Noah Wardrip-Fruin, and **Michael Mateas**

Jack Baskin School of Engineering
University of California at Santa Cruz
1156 High Street, Santa Cruz, CA 95064
{batman, nwf, michaelm}@cs.ucsc.edu

Abstract

We are recreating, investigating, and defining new uses for one of the most influential artificial intelligence projects of the past 25 years: Scott Turner's Minstrel, which is regarded as a landmark in both the story generation and computational creativity communities. We compare our new system, Minstrel Remixed, with the implementation of the original, and discuss the various additions made during our rational reconstruction which facilitate investigations into the inner workings of the system. In conclusion, we evaluate the performance of Minstrel Remixed and determine that its results are quite close to those of the original.

Introduction

While ongoing progress in digital entertainment technology continues, commercial designers still largely eschew systems for procedural story generation, preferring instead to generate content by hand. In the academic literature, projects such as (Appling & Riedl 2009, Roberts & Isbell 2009) continue to investigate ways to improve the nuances of interactive storytelling while others attempt to create their own systems to investigate ways to use knowledge from interactive narrative and story generation in new fields such as playable games (Drachen & Hitchens et al. 2009, Sullivan, Mateas & Wardrip-Fruin 2009). As a complement to current research in the field, revisiting the landmark systems of the 1970s and 80s with modern computers and techniques may yield fruitful results. One such landmark is Minstrel, developed by Scott Turner for his Ph.D thesis (Turner 1993). Despite the fact that Minstrel saw no further testing or investigation beyond that which Turner performed in his dissertation, it is still one of the most acclaimed story generation systems to date.

Since a working copy of Minstrel does not exist to be investigated and the system was designed and implemented so long ago, Minstrel is an attractive candidate for rational reconstruction (see below) followed by thorough testing.

In the ongoing reconstruction of Minstrel we investigate several topics, including: what authorial affordances Minstrel provides, what new representations might be needed to expand the system beyond strictly passive story generation, and what new algorithms might be exploited to favorably alter its performance. Through answering these questions and continuing to explore what can be done with such a successful system from the past, we have discovered many interesting possible uses for Minstrel and a number of insights into how and why it works.

Related Work

This section briefly outlines previous work done in the three principal areas relevant to this project: classical story generation systems, case based reasoning, and rational reconstruction.

Classical Story Generation. The first major story generation system, which preceded Minstrel and which also received significant attention, is Tale-Spin (Meehan 1977). Like Minstrel, this system generates stories which satisfy user-submitted requirements. Tale-Spin creates English stories by planning a method for the main character to achieve her or his goal, using inferences and rules to generate a large number of details about a story (many of which do little contribute to an audience experience). This contrasts nicely with Minstrel, which performs no logical inferences and which performs all actions from the point of view of an author, manipulating all parts of the story in parallel.

Along with Minstrel and Tale-Spin, the other widely-discussed early story generator is Universe (Lebowitz 1985) later reconstructed as WideRuled (Skorupski, Jayapalan, et al. 2007) which implement a Hierarchical Task Network planner to generate stories. These systems, like Minstrel, generate stories from an author's perspective. Unlike Minstrel's creative approach, however, they are organized around immutable "plot fragments."

Case Based Reasoning. Minstrel was developed as an approach to creativity for Case Based Reasoning (CBR) and its approach has been followed up in some past work, some rather far afield. For example, TRAM-like operators were applied to feature vector-based case representations in the real-time strategy game Wargus in multiple studies (Weber & Mateas 2009, Aha 2005).

A bit closer to the original Minstrel, multiple projects have been published in which Vladimir Propp's work on story fragment interchangeability (Propp 1968) is leveraged to assist CBR systems to dynamically generate

stories. One such such system was created to manage a multiplayer game environment (Fairclough 2003) while another has the same goals as Minstrel but uses predefined relevant knowledge to supplement its CBR instead of applying transformations (Gervas 2007).

Rational Reconstruction. The normal goal of rational reconstruction is to investigate the inner workings of a system by altering some components of the system and comparing the results to the original. Once accomplished, arguments can then be made about the pieces that were altered and a view of the potential of the underlying system can be separated from the arbitrary programming that the original author implemented.

A number of projects (Musen, Gennarj & Wong 1995, Tate 1995) have successfully used rational reconstruction to better understand the fundamental concepts of the original systems and to explore various improvements. It should also be noted that a partial reconstruction of Minstrel was performed (Peinado 2006) in which the knowledge representation systems of Minstrel were recreated in W3C's OWL. While this did a good job of proving that the knowledge representation can be successfully recast, without the rest of the system in place it is far from a complete reconstruction.

Minstrel Remixed

The original Minstrel was created to explore how well human creativity could be simulated in the space of story generation while taking advantage of the Case Based Reasoning theories of the time (Turner, 1994). Our reconstruction, Minstrel Remixed (MR), is a useful but necessarily imperfect recreation based on written descriptions of the original Minstrel—since neither the source nor a working copy is available today. Our three goals for MR differ from those of the original: to identify elements of the original which were crucial to its operation, to explore possible uses for the system outside of its original scope, and to provide a version of Minstrel to the research community for analysis and general use.

Architecture

Like Turner's Minstrel, MR can be broadly broken down into two main components. The Transform Recall Adapt Method (TRAM) system is a case based reasoner which modifies story details. Working above the TRAM system is the Author Level Planning system which enforces constraints and improves stories as they are generated.

Both systems use story subgraphs as their primitives. Figure 1 shows the high level progression of requests that make up a single step of the story generation loop. First an Author Level Plan (ALP) notes that there is a chunk of the story that is underspecified or in some way requires attention (e.g., a character appears without an introduction, major events appear without dramatic support, etc.). This causes a call to the TRAM system asking for a story detail with some specific characteristics. Once that is found or pieced together from the story library, the results are adapted back to modify the current story.

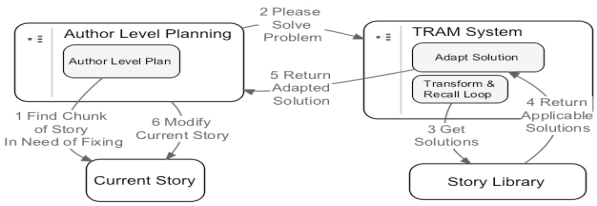

Figure 1. Overview of Minstrel's Components

Story Graphs. Stories in MR are represented exactly like those in the original Minstrel, as graphs with labeled directed edges and a number of facts attached to each node. There are four types of nodes: state, act, goal, and belief. Each of these can have values assigned to the following fields: type, actor, object, value, to, and from. When combined with appropriate edges and a list of nouns, these graphs have a one-to-one relationship with completely realized stories. An example story graph can be seen in Figure 2, which illustrates a very short story describing a princess who drinks a potion in order to injure herself.

TRAMs. (Transform Recall Adapt Methods) perform all of the fine-grained story graph edits in both Minstrel and Minstrel Remixed. TRAMs are small bundles of operations designed to help recall useful information from the story library. Each TRAM takes a requirements graph as input and performs a transformation to produce a new graph. It then performs a recall from the story library based on the new graph and attempts to adapt the matches back to the original query through a reverse transformation. There are many TRAMs in the library because each one has its own unique transformation and reverse transformation, giving the system flexibility. Just as in Turner's original work, the TRAM system in its simplest mode of operation is given a requirements graph which it uses to perform straight recall from the story library. But the creative power of the TRAMs is their ability to transform the requirements graph into something different and then adapt the results back to a useable story graph. This transformation and adaptation is done through one or more of a library of individual TRAMs. The most generic of these TRAMs is called Generalize Constraint—which simply takes the requirements graph and makes it more general by removing one of the constraints.

An example of TRAMs in action is as follows: a graph is passed in requiring a knight, John, to die by the sword. By transforming this initial query using a TRAM called Generalized Constraint, we might end up with a resulting requirement in which a something dies by the sword. If this then matched to a story about another knight, Frances who has a duel with an Ogre and kills it, then the TRAM system could replace the Ogre with John the knight and return a story fragment about a duel between John and Frances in which John dies.

The creative power of TRAMs ultimately comes from their ability to find cases in the story library which aren't easily recognizable as applicable. In the original Minstrel, a search which returns no results will be transformed by a random TRAM and then restarted. This often leads to a random sequence of transformations being applied to the search term before a result is located. In MR however, we have enabled TRAMs to be chosen intelligently, oftentimes resulting in fewer transformations being needed to get search results and thus more similarity between the original query and the eventual result. To illustrate the ability that the TRAM approach has to create truly unexpected results, we can look at other ways for John to die. Let's say that the first TRAM applied is Similar Outcomes Partial Change, which changes the death requirement of John's story to some ambiguous change in health. Let's also say that, when this failed to match anything in the story library, the Generalize Noun TRAM was triggered next, which generalized John to a Generic person. These changes allowed the requirements to match with the story fragment from Figure 2 in which John matches to Princess Peach who uses a potion to hurt (rather than kill) herself. Upon adaptation back to the current requirements, the resulting story is that John commits suicide, killing himself dramatically with a potion (See Figure 3 for the whole progression).

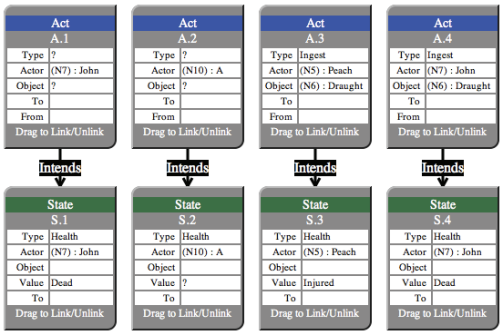

Figure 3. TRAM progression to suicide.

Although the original TRAM system functioned well and created interesting results for Minstrel to work with, during our reconstruction we decided that the TRAM system was a ripe place to investigate possible changes. So we included hooks in the TRAM system of Minstrel Remixed which enable programmers to change the manner in which TRAMs are selected and applied. Two selection algorithms which are discussed below in the rational reconstruction section of this paper have been included and can be selected from by toggling a single variable.

Generalized Story Templates. Planning Advice Themes (PATs) are parable based story templates which are used to start stories. The original Minstrel exclusively used PATs and could produce stories to illustrate themes such as "a bird in the hand is worth two in the bush." In MR we have opted for a more generalized target and thus have a wide array of templates, some of which do not describe adages. Although they are the same structurally, we call these more generic PATs Generalized Story Templates (GSTs). Because all PATs can be translated into the more flexible GSTs with little or no work, this gives MR more overall flexibility in its story frameworks. GSTs have the same structure as a completed story graph but they are generally full of holes or placeholder variables that need to be filled in. An example GST is shown in Figure 4, in which a person asks another person for help and successfully solicits their aid.

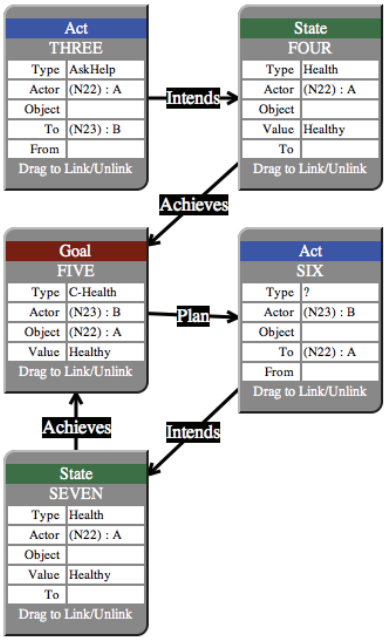

Figure 4. Extremely Simple GST

Because the structure of the GST provides an outline that is filled in by the other systems in Minstrel Remixed, the quality of the GSTs is directly linked to the quality of the resulting stories. As the very first step in any story creation in MR, any given requirements are used to select an appropriate GST, which is then installed into the story graph for the rest of the system to work with.

Author Level Plans. The Author Level Planning system contains a number of Author Level Plans (ALPs) which guide story construction at a high level. ALPs are responsible for looking at the current state of their target (the current story graph) and planning modifications to it with the eventual goal of filling in the whole story graph in a desirable manner. There are three classes of ALP which operate together: Story Producers, Story Checkers, and Story Enhancers. Producers are the simplest form of ALP and operate by handing story subgraphs that have blank variables to the TRAM system to be filled out. The Checkers are only slightly more complex in that they each search over the story graph for very specific subgraphs and add other ALPs to the queue in order to deal with the discovered subgraphs. Enhancers are used to add rich characteristics to a story such as tragedy, suspense, and characterization. As such, enhancers will often add

additional details to the story outside of its original bounds, generally in the form of new nodes being added to the story graph.

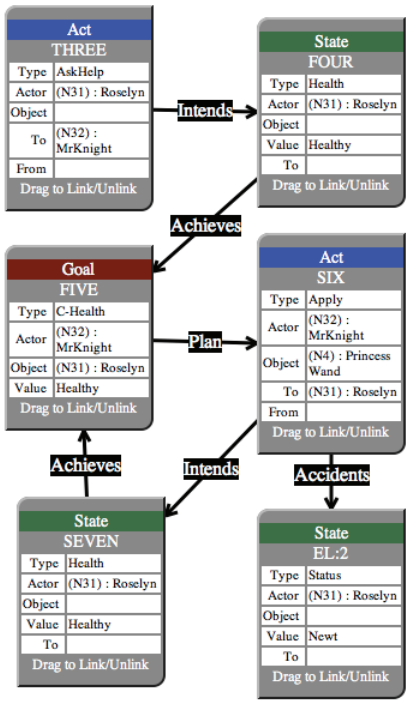

Figure 5. A Completely Generated Story

Figure 5 shows the ALP System in action, starting with an empty story graph in which person A asks person B for help in becoming more healthy. The first ALP in the queue that activates in this case will be a Completeness Checker which looks for question marks or undefined nouns (in this case A and B). The Completeness Checker, upon finding the empty variables in the story graph, will put the Story Producer ALP in the queue with its target set to the nodes in question. The Story Producer ALP targets 'Act SIX' with its question marked entry and pulls 'Goal FIVE' and 'State SEVEN' in as references and passes the trio of nodes off to the TRAM system to get a matching story fragment. It gets back a story in which MrKnight uses a magical Princess Wand on Roselyn to make her healthy. In cleaning up, the producer ALP changes all instances of A and B in the story to match Roselyn and MrKnight and returns the story shown in figure 5. Once that is complete, a checker ALP called Accidental Consequences runs over the story in search of acts which have outgoing intends edges but no outgoing accidents edges. This checker is designed to find places where Story Enhancers can be brought to bear to make the story more interesting or detailed. In this case it's trying to find good candidates for side effects of main plot actions which can be either left alone to provide background details or which can later be woven back into the plot. It notes two such places and queues an enhancement ALP which decides that the extra state, EL:2 should be added to enhance the story by adding an accidental side-effect of the princess wand. Although in this instance MR was stopped before continuing on from this point, additional enhancement could have been brought to bear to further embellish upon the completed story.

Modernization. Although Minstrel Remixed attempts to stick to the original designs for Minstrel as much as possible, improvements have been added to make it easier to use in a modern setting. MR is coded in Scala which can operate as a functional language like Minstrel's LISP variant but can compile down to Java source or Java byte code. Additionally, MR will read XML files into its story library, scripts, and GSTs. This makes it much easier to author new content, swap entire story libraries in and out, and exchange specific stories between instances. We also included an interactive text based shell and an output routine for story graphs which allows for visual representations to be generated as PDFs.

Minstrel's Rational Reconstruction

Traditional rational reconstruction is performed in order to investigate the importance of various components, concepts, and algorithms to the functionality of a piece of software. Through studies of this sort we are able to learn what is crucial to the underlying operation of the software and what is merely an implementation detail. We have kept these goals in mind while building Minstrel Remixed. Turner's dissertation (Turner 1993) was used as a starting point for MR but in keeping with the goals of a rational reconstruction we have sought alterations that might give some clues as to the full potential of the system. As a result, during our reviewing of the code base we have supplemented the original functionality in a number of areas to provide insights into what makes Minstrel unique. While the focus of the project has been on a faithful recreation, in this section we describe the algorithm-level changes and alternate applications of Minstrel that we are exploring. By rationally reconstructing Minstrel, we enable these explorations of Minstrel's generative model.

In Minstrel the TRAMs are a crucial aspect of the functioning of the system. TRAMs function by being given a requirement which they then transform and attempt to match. Minstrel implemented an index tree that spanned the story library in order to quickly retrieve only those cases which were relevant for matching. MR deviates from the original functionality in that it uses the increased processing power of modern machines to find all subgraphs in the story library which have the same types of nodes (i.e., State, Act, Goal, Belief) in the same order as the requirement graph and then attempts to match against them all. Additionally, where Minstrel only has one TRAM application strategy, a depth-limited depth-first search of sequentially applied random transforms, MR was implemented with a hook upon which any appropriate algorithm can be mounted and toggled on or off. As can be seen in figure 6, Minstrel's random TRAM application (zig zag arrows) is inefficient and often requires more

transformation to return results than the optimum transformation. Although transformations are directly correlated with perceived creativity of results, more transformations also tends to lead to a higher likelihood of incompatible returns, leading to nonsensical story fragments. In contrast to the random selection method, the currently implemented graph distance algorithm (the direct line from origin to closest point on figure 6), is able to very efficiently ascertain which TRAMs would be required in order to match the requirements graph to the graph in question. By giving a penalty value to each TRAM and applying this algorithm to all possible matches, MR is provided with a sorted list of the closest matches (lowest penalties) from which it can choose and then perform the proscribed TRAMs to retrieve a solution. With some random weighting applied to prevent MR from always selecting the closest match, this method approximates the capabilities of the random method without deviating so far from ideal that nonsensical results are returned.

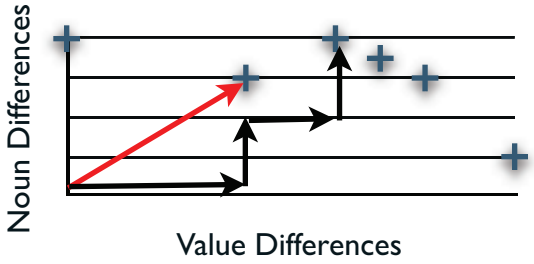

Figure 6. TRAM Matching. Minstrel's original random TRAM selector (zig-zag) compared to graph distance TRAM selector (slanted path).

Interactivity. An interesting potential change between Minstrel and Minstrel Remixed is made possible by the dramatic improvement in processing power that has taken place in the past two decades. While MR has not been tested with thousands of stories in its library to search over, initial tests of story creation happen instantaneously rather than in tens of minutes (as was the case for Minstrel). Using the new speed inherent in MR, interactivity can be achieved in realtime. Although the thrust of the current work has been towards recreating a functioning version of Minstrel, care was taken to enable new functionality in the ALP system such that all actions that are taken are recorded in a manner that enables the system to roll back story creation to a specific point and continue construction along a different path. We have successfully performed proof of concept tests in which a story has been generated, rolled back to a specific point, modified by a user, and then regenerated, incorporating the story prefix (the part that has already happened) and the modification. The potential to support interaction opens a number of research directions that were previously closed to the Minstrel system.

Evaluation

Since the original Minstrel is not available for comparison, the only evaluative steps possible are to look at the examples included in Turner's book and dissertation. When compared with these examples, MR appears to work as intended. One example of a successful creative transformation and recall by the original Minstrel is given in figure 2. It was later reproduced by MR. Many of Turner's other examples were tested and shown to work as well, including one example which has been often used to demonstrate one of the major flaws with Minstrel in which a knight attempts to curry favor with a princess by giving her a hunk of meat (created by transforming a story about a knight giving a hunk of meat to a dragon in order to appease it). Although there are obvious reasons for this outcome, it demonstrates that MR shares its progenitor's limitation of not understanding any of the common sense of the worlds behind the stories that it tells. It's reasonable to assume that both dragons and princesses like many of the same things (gold, gems, etc.) but there's no way for MR to tell whether a gift for a dragon, horse, or hermit would be of any interest to a princess. There is obvious potential for work to be done in this area and we plan to investigate means of providing some domain-appropriate common sense knowledge MR as needed for projects in the future.

Although we have no way of knowing how many story fragments were incorporated into the original, we do know that Minstrel's story library was roughly equivalent to the content of a few short stories. MR now has forty stories of varying length (in two domains) along with eleven of the original twenty-four TRAMs. The current contents of the system have proved rich enough to procedurally generate many viable story fragments from the original Arthurian domain that Turner used. For completeness' sake, we have included a new domain of storytelling that revolves around conspiracy theories. In this domain we had MR generate 150 story fragments and of those, some (ten to twelve) appear to be interestingly unexpected (e.g., assassination by alien abduction) and only 7 appear nonsensical. While we acknowledge that a fantastic domain such as conspiracy theories lowers the bar for fragments being useful, these results indicate that MR is capable not only of reproducing much of the original results of Minstrel but also of generating stories in a completely novel domain.

Future Work

There are many future challenges and opportunities for Minstrel Remixed. To begin with, problems due to MR's inability to understand common sense in its domain (such as princesses being gifted meat) are unacceptable for many uses. We believe that integrating some form of knowledge base (such as ConceptNet (Liu & Singh 2004)) for use in the TRAMs might allow us to improve the results of the TRAMs without impacting the creative space too greatly.

Additionally, the story library for MR is currently small. An authoring tool is currently in development, which allows users to rapidly create stories and fill MR's library. By use of this tool, we hope to flesh out the Arthurian and Conspiracy domains and enable other users to create and rapidly fill their own domain libraries as well. In addition to filling the story library, a secondary task that needs to be accomplished is to implement a system by which MR can output English text in a manner equivalent to that employed by the original Minstrel. A prototype for this system is currently in production, but more work is needed before it will be helpful in translating MR's graphs into English.

Aside from connecting to common sense reasoning, work to fill out the story library, and developing the ability to report stories in English, additional rational reconstruction would still be fruitful. Although hooks were embedded into the TRAM system to allow us to investigate the effects of alternate searching algorithms, additional investigations could no doubt be made into the ALP system. Finally, we believe that there are many ways in which MR could be made to be interactive.

Conclusion

This paper has described Minstrel Remixed, a reconstructed and working version of the original Minstrel system created by Scott Turner. We have discussed a number of improvements to the original design as well as some of the potential applications for Minstrel Remixed which lie well outside of the originally intended uses of Minstrel. We believe that future work on Minstrel Remixed will provide interesting insights into the nature of the creativity demonstrated by the TRAM system, the flexibility of the system as a whole, and the utility that Minstrel Remixed will have in Interactive Narrative applications.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0747522.

References

Appling, D. and Riedl, M. 2009. Representations for Learning to Summarize Plots. In *Proceedings of the 2009 AAAI Symposium on Intelligent Narrative Technologies II*.

Drachen, A., Hitchens, M., Aylett, R., and Louchart, S. 2009. Modeling Game Master-based story facilitation in multi-player Role-Playing Games. In *Proceedings of the 2009 AAAI Symposium on Intelligent Narrative Technologies II*, 24–32.

Fairclough, C. and Cunningham, P. 2003. A multiplayer case based story engine. In *4th International Conference on Intelligent Games and Simulation*, 41–46.

Gervas, P., Diaz-Agudo, B., and Hervas, R. Story plot generation based on cbr. *Applications and Innovations in Intelligent Systems XII*, 33–46.

Lebowitz, M. 1985. Story-telling as planning and learning. *Poetics*, 14(6).

Liu, H. and Singh, P. 2004. ConceptNet—a practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):211–226.

Meehan, J. 1977. Tale-spin, an interactive program that writes stories. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*.

Musen, M., Gennari, J., and Wong, W. 1995. A rational reconstruction of INTERNIST-I using PROTEGE-II. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*

Peinado, F., Gervas P. 2006. Minstrel reloaded: from the magic of lisp to the formal semantics of OWL. in *Technologies for Interactive Digital Storytelling and Entertainment*, 93–97.

Propp, V. 1968. Morphology of the Folktale, trans. Laurence Scott (Austin: University of Texas Press).

Roberts, D., Narayanan, H., and Isbell, C. 2009. Learning to Influence Emotional Responses for Interactive Storytelling. In *Proceedings of the 2009 AAAI Symposium on Intelligent Narrative Technologies II*.

Skorupski, J., Jayapalan, L., Marquez, S., and Mateas, M. 2007. Wide ruled: A friendly interface to author-goal based story generation. *LECTURE NOTES IN COMPUTER SCIENCE*, 4871:26.

Sullivan, A., Mateas, M., and Wardrip-Fruin, N. 2009. QuestBrowser: Making Quests Playable with Computer-Assisted Design.

Tate, A. 1995. Integrating Constraint Management into an AI Planner. *Artificial Intelligence in Engineering*, 9(3): 221–228.

Turner, S. (1993). MINSTREL: a computer model of creativity and storytelling.

Turner, S. (1994). The creative process: a computer model of storytelling and creativity.

W. Aha, D., Molineaux, M., and Ponsen, M. 2005. Learning to win: Case-based plan selection in a real-time strategy game. *Case-Based Reasoning Research and Development*, 5–20.

Weber, B. and Mateas, M. 2009. Conceptual Neighborhoods for Retrieval in Case-Based Reasoning. In *Proceedings of the 8th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, 357.