Case-Based Agents

Workshop at the
Twenty-Third International Conference on
Case-Based Reasoning
(ICCBR 2015)

Frankfurt, Germany
September 2015

David W. Aha and Michael W. Floyd (Editors)

Chairs

David W. Aha	Naval Research Laboratory, USA
Michael W. Floyd	Knexus Research Corporation, USA

Program Committee

Klaus-Dieter Althoff	University of Hildesheim, Germany
Daniel Borrajo	University of Carlos III of Madrid, Spain
Sutanu Chakraborti	IIT Madras, India
Alexandra Coman	Ohio Northern University, USA
Amélie Cordier	Claude Bernard University of Lyon 1, France
Santiago Ontañón	Drexel University, USA
Miltos Petridis	University of Brighton, UK
Jonathan Rubin	Palo Alto Research Center, USA
Swaroop Vattam	NRC and Naval Research Laboratory, USA

Additional Reviewers

Pascal Reuss	University of Hildesheim, Germany
Viktor Ayzenshtadt	University of Hildesheim, Germany

Preface

The ICCBR 2015 Workshop on Case-Based Agents aims to highlight the challenges autonomous agents encounter and how case-based reasoning (CBR) can be used to overcome those challenges (e.g., complex environments, imperfect information, interacting with both teammates and adversaries, unexpected events). We believe a natural synergy exists between CBR and agents, and hope this workshop will highlight the recent progress in both disciplines. This serves as a sequel to the first Workshop on Case-Based Agents, which was held at ICCBR 2014 in Cork, Ireland.

The workshop program includes eight papers that explore various ways agents can leverage case-based reasoning. Two of the papers examine agents that can detect faults or discrepancies and identify their root cases. Reuss et al. explore multi-agent fault diagnosis in an aircraft domain whereas Kann et al. discuss the KRePE system, which identifies discrepancies while performing naval mine countermeasure missions. Gabel and Godehardt use CBR to predict an opponents low-level actions in simulated robotic soccer. Similarly, Frazer et al. present an error-tolerant plan matching algorithm to improve the performance of case-based plan recognition.

Fitzgerald and Goel describe their ongoing work in robotic learning by demonstration, with a specific focus on case storage and adaptation. Sánchez-Ruiz also presents an agent that observes other agents, but instead of learning to perform an observed behavior it learns to predict the outcome of battles in StarCraft. Paul and Huellermeier describe an agent that plays the Angry Birds game. The agent learns actions to perform using random exploration and stores cases containing the best action for each encountered game state. Coman et al. propose an agent that can potentially have conflicts between its own goals and motivations, and the goals or plans supplied to it by a user. This can cause situations where the agent might rebel rather than blindly follow user commands.

Overall, we believe these papers provide a good sampling of the ways in which case-based reasoning has been used by agents and highlight recent research trends in this area. We hope that this workshop will both provide a venue for researchers in this area to meet and discuss their work, as well as provide an entry point for researchers interested in learning about case-based agents. We would like to thank everyone who contributed to the success of this workshop, including the authors, program committee, reviewers, and the ICCBR 2015 conference organizers.

September 2015 — David W. Aha
Frankfurt — Michael W. Floyd

I Know What You're Doing: A Case Study on Case-Based Opponent Modeling and Low-Level Action Prediction

Thomas Gabel and Eicke Godehardt

Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
60318 Frankfurt am Main, Germany
{tgabel|godehardt}@fb2.fra-uas.de

Abstract. This paper focuses on an investigation of case-based opponent player modeling in the domain of simulated robotic soccer. While in previous and related work it has frequently been claimed that the prediction of low-level actions of an opponent agent in this application domain is infeasible, we show that – at least in certain settings – an online prediction of the opponent's actions can be made with high accuracy. We also stress why the ability to know the opponent's next low-level move can be of enormous utility to one's own playing strategy.

1 Introduction

Recognizing and predicting agent behavior is of crucial importance specifically in adversary domains. The case study presented in this paper is concerned with the prediction of the low-level behavior of agents in the highly dynamic, heterogeneous, and competitive domain of robotic soccer simulation (RoboCup). Case-based reasoning represents one of the potentially useful methodologies for accomplishing the analysis of the behavior of a single or a team of agents. In this sense, the basic idea of our approach is to make a case-based agent observe its opponent and, in an online fashion, i.e. during real game play, build up a case base to be used for predicting the opponent's future actions.

In Section 2, we introduce the opponent modeling problem, point to related work, and argue why knowing an opponent's next low-level actions can be beneficial. The remainder of the paper then outlines our case-based methodology (Section 3), reviews the experimental results we obtained (Section 4), and summarizes and discusses our findings (Section 5).

2 Opponent Modeling in Robotic Soccer Simulation

RoboCup [12] is an international research initiative intending to expedite artificial intelligence and intelligent robotics research by defining a set of standard problems where various technologies can and ought to be combined solving them. Annually, there are championship tournaments in several leagues – ranging from rescue tasks over real soccer-playing robots to simulated ones.

In Proceedings of the ICCBR 2015 Workshops. Frankfurt, Germany.

2.1 Robotic Soccer Simulation

The focus of the paper at hand is laid upon RoboCup's 2D Simulation League, where two teams of simulated soccer-playing agents compete against one another using the Soccer Server [10], a real-time soccer simulation system.

The Soccer Server allows autonomous software agents written in an arbitrary programming language to play soccer in a client/server-based style: The server simulates the playing field, communication, the environment and its dynamics, while the clients – eleven autonomous agents per team – connect to the server and are permitted to send their intended actions (e.g. a parameterized kick or dash command) once per simulation cycle to the server via UDP. Then, the server takes all agents' actions into account, computes the subsequent world state and provides all agents with (partial) information about their environment via appropriate messages over UDP.

So, decision making must be performed in real-time or, more precisely, in discrete time steps: Every 100ms the agents can execute a low-level action and the world-state will change based on the individual actions of all players. Speaking about low-level actions, we should make clear that the actions themselves are "parameterized basic actions" and the agent can execute only one of them per time step:

- $dash(x, \alpha)$ – lets the agent accelerate along its current body orientation by relative power $x \in [0, 100]$ (if it does not accelerate, then its velocity decays) into direction $\alpha \in (-180, 180]$ relative to its body orientation
- $turn(\alpha)$ – makes the agent turn its body by $\alpha \in (-180, 180]$ where, however, the Soccer Server reduces α depending on the player's current velocity in order to simulate an inertia moment
- $kick(x, \alpha)$ – has an effect only, if the ball is within the player's kick range (1.085m around the player) and yields a kick of the ball by relative power $x \in [0, 100]$ into direction $\alpha \in (-180, 180]$
- There exist a few further actions (like tackling[1], playing foul, or, for the goal keeper, catching the ball) whose exact description is beyond scope.

Given this short description of the most important low-level actions that can be employed by the agent, it is clear that these basic actions must be combined cleverly in consecutive time steps in order to create "higher-level actions" like intercepting balls, playing passes, doing dribblings, or marking players. We will call those higher-level actions *skills* in the remainder of this paper.

Robotic Soccer represents an excellent testbed for machine learning, including approaches that involve case-based reasoning. For example, several research groups have dealt with the task of learning parts of a soccer-playing agent's behavior autonomously (for instance [9, 8, 3]). In [6], as an other example, we specifically addressed the issue of using CBR for the development of a player agent skill for intercepting balls.

---

[1] To tackle for the ball with a low-level action $tackle(\alpha)$ means to straddle for the ball and thus changing its velocity, even if it is not in the player's immediate kick range; such an action succeeds only with limited probability which decreases the farther the ball is away from the agent.

2.2 Related Work on Opponent Modeling

Opponent modeling is an important factor that can contribute substantially to a player's capabilities in a game, since it enables the prediction of future actions of the opponent. In doing so, it also allows for adapting one's own behavior accordingly. Case-based reasoning has been frequently used as a technique for opponent modeling in multi-agent games [4], including the domain of robotic soccer [13, 1].

Using CBR, in [13] the authors make their simulated soccer agents recognize currently executed higher-lever behaviors of the currently ball leading opponent player. These include passing, dribbling, goal-kicking and clearing. These higher-level behaviors correspond to what we refer to as skills, i.e. action sequences that are executed over a dozen or more time steps. This longer time horizon allows the agent to take appropriate counter measures.

The authors of [11] also deal with the case-based recognition of skills (higher-level behaviors, to be exact the shoot-on-goal skill) executed by an opponent soccer player, focusing on the appropriate adjustment of the similarity measure employed. While we do also think opponent modeling is useful for counteracting adversary agents, we, however, disagree with these authors claiming that "in a complex domain such as RoboCup it is infeasible to predict an agent's behavior in terms of primitive actions". Instead we will show empirically that such a low-level action prediction can be achieved during an on-going play using case-based methods. To this end, the work presented in this paper is also related to the work by Floyd et al. [5] whose goal is to mimic the overall behavior of entire soccer simulation teams, be it for the purpose of analysis or for rapid prototyping when developing one's own team, without putting too much emphasis on whether the imitators yield competitive behavior.

2.3 Related Previous Work

What is the use of knowing exactly whether an opponent is going to execute a $kick(40, 30°)$ or a $dash(80, 0°)$ low-level action next? This piece of information certainly does not reveal whether this opponent's intention is to play a pass (and to which teammate) in the near future or to dribble along. Clearly, for answering questions like that the approaches listed in the previous section are potentially more useful. But knowing the opposing agent's next low-level actions is extremely useful, when knowing the *next state on the field* is essential (cf. Figure 1 for an illustration).

In [7], we considered a soccer simulation defense scenario of crucial importance: We focused on situations where one of our players had to interfere and disturb an opponent ball leading player in order to scotch the opponent team's attack at an early stage and, even better, to eventually conquer the ball initiating a counter attack. We employed a reinforcement learning (RL) methodology that enabled our agents to autonomously acquire such an aggressive duel behavior, and we successfully embedded it into our soccer simulation team's defensive strategy. So, the goal was to learn a so-called "duelling skill" (i.e. a higher-level behavior which in the end yields a sequence of low-level actions) which made our agent conquer the ball from the ball-leading opponent.

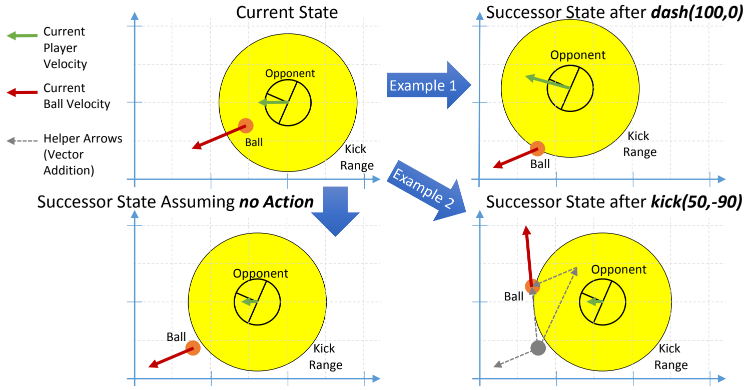

Current State
Current Player Velocity
Current Ball Velocity
Helper Arrows (Vector Addition)
Opponent
Ball
Kick Range
Example 1
Example 2
Successor State after ***dash(100,0)***
Successor State Assuming ***no Action***
Successor State after ***kick(50,-90)***

Fig. 1. In the top-left we see the current state of an opponent agent in ball possession. If we assume, this agent does not take any low-level action, then the resulting successor state looks like the one in the bottom left figure: Player and ball have moved according to their recent velocities while the magnitude of the velocity vectors have decayed according to the rules of the simulation. How different the successor state may look, if the opponent, however, does take an action (which is most likely), is shown in the right figures. In example 1 (top) the agent accelerates full power along its current body orientation, while the ball is not affected. In example 2, the player kicks the ball with 50% power into -90° relative to its current body orientation which yields a resulting ball velocity vector as shown in the bottom right.

An important feature of the soccer simulation domain is that the *model* of the environment is known. This means given, for example, the current position and velocity of the ball, it is possible for any agent to calculate the position of the ball in the next time step (because the implementation of the physical simulation by the Soccer Server is open source[2]). As a second example, when knowing one's own current position, velocity and body angle, and issuing a $turn(68°)$ low-level action, the agent can precalculate the position, velocity and body orientation it will have in the next step. Or, finally, when the agent knows the position and velocity of the ball, it can precalculate the ball's position and velocity in the next step, for any $kick(x, \alpha)$ command that it might issue.

Knowing the model of the environment (formally, the transition function $p : S \times A \times S \to \mathbb{R}$ where $p(s, a, s')$ tells the probability to end up in the next state s' when executing action a in the current state s), is extremely advantageous in reinforcement learning, since then model-based instead of model-free

---

[2] In practice, the Soccer Server adds some noise to all low-level actions executed, but this is of minor importance to our concerns.

learning algorithms can be applied which typically comes along with a pleasant simplification of the learning task.

So, in soccer simulation the transition function p (model of the environment) is given since the way the Soccer Server simulates a soccer match is known. In the above-mentioned “duelling task”, however, the situation is aggravated: Here, we have to consider the influence of an opponent whose next actions cannot be controlled. In [7], we stated that the opponent’s next (low-level) actions can “hardly be predicted [which] makes it impossible to accurately anticipate the successor state”, knowing which is, as pointed out, extremely useful in RL. In the paper at hand, we will show that predicting the opponent’s next low-level action might be easier than expected. As a consequence,

- in [7] we had to rely on a rough approximation of p, that merely takes into account that part of the state that can be influenced directly by the learning agent and which ignored the part of the future state which is under direct control of the ball-leading opponent (e.g. the position of the ball in the next state). This corresponded to the unrealistic assumption of an opponent that never takes any action (cf. Figure 1, bottom left).
- in future work we can employ a much more accurate version of p based on the case-based prediction of the opponent’s low-level actions described in the next section.

3 Case-Based Prediction of Low-Level Actions

In what follows, we differentiate between an opponent (OPP) agent whose next low-level actions are to be predicted as well as (our) case-based agent (CBA) that essentially observes the opponent and that is going to build up a case base to be used for the prediction of OPP’s actions.

When approaching the opponent modeling problem as a case-based reasoning problem, the goal of the case-based agent is to correctly predict the next action of its opponent given a characterization of the current situation. Stated differently, the current state of the system (including the case-based agent itself, its opponent as well as all other relevant objects) represents a new query q. CBA’s case base $\mathcal{C}$ is made up of cases $c = (p, s)$ whose problem parts p correspond to other, older situations and corresponding solutions s which describe the action OPP has taken in situation p. Next, the case-based agent will search its case base for that case $\hat{c} = (\hat{p}, \hat{s}) \in \mathcal{C}$ (or for a set of k such cases) whose problem part features the highest similarity to the current problem q and employ its solution $\hat{s}$ as the current prediction of the opponent’s next action.

3.1 Problem Modeling

In the context of this case study we focus on dribbling opponents, i.e. the opponent has the ball in its kick range and moves along while keeping the ball within its kick range all the time. Stated differently, we focus on situations where OPP behaves according to some "dribble skill" (a higher-level dribbling behavior). Consequently, OPP executes in each time step one of the three actions $kick(x, \alpha)$, $dash(x, \alpha)$, or $turn(\alpha)$. The standard rules of the simulation allow x to be from $[0, 100]$ and α from $(-180°, 180°]$ for kicks and turns. For dashes, α is allowed to take one out of eight values (multiples of 45°). In almost all cases occurring during normal play, however, a dribbling player is heading more or less towards his opponent's goal which is why the execution of low-level turn actions represents an exceptional case. Therefore, for the time being, we leave turn actions aside and focus on the correct prediction of dashes and kicks including their parameters x and α.

Case Structure The state of the dribbling opponent (OPP) can be characterized by the x and y position of the ball within its kick range ($pos_{b,x}$ and $pos_{b,y}$) relative to the center of OPP as well as the x and y components of the ball's velocity ($vel_{b,x}$ and $vel_{b,y}$; of course, these values are also relative to OPP's body orientation). Moreover, OPP's x and y velocities ($vel_{p,x}$ and $vel_{p,y}$) are of relevance, making six features in total. The seventh relevant feature, OPP's current body orientation θ_p can be skipped due to the arguments mentioned in the preceding paragraph. Furthermore, the y component of OPP's velocity vector $vel_{p,y}$ is, in general, zero since a dribbling player almost always dribbles along its current body orientation. While this allows us to also skip the sixth feature, we remove a redundancy in the remaining features (and thus arrive at only four of them) by changing to a relative state description that incorporates some background knowledge[3] from the simulation. Hence, the problem part p of a case $c = (p, s)$ is a four-tuple $p = (pos_{bnx}, pos_{bny}, vel_{bnx}, vel_{bny})$ with

$$\begin{aligned} pos_{bnx} &= pos_{b,x} + 0.94 \cdot vel_{b,x} - 0.4 \cdot vel_{p,x} \\ pos_{bny} &= pos_{b,y} + 0.94 \cdot vel_{b,y} - 0.4 \cdot vel_{p,y} \\ vel_{pnx} &= 0.94 \cdot vel_{b,x} - 0.4 \cdot vel_{p,x} \\ vel_{pny} &= 0.94 \cdot vel_{b,y} - 0.4 \cdot vel_{p,y} \end{aligned}$$

where all components characterize the next state as it would arise, if the agent would not take any action (cf. Figure 1).

The solution s of a case $c = (p, s)$ consists of a class label l ("dash" or "kick") as well as two accompanying real-valued attributes for the power x and angle α parameters of the respective action. Thus, the solution is a triple $s = (l, x, \alpha)$.

3.2 Implementing the CBR Cycle

The case-based agent CBA observes his opponent OPP and, in doing so, builds up its case base. Note that all agents in soccer simulation act on incomplete and uncertain information. Their visual input consists of noisy information about objects in their limited field of vision. However, if the observed opponents are

---

[3] Knowledge about how the Soccer Server decays objects.

near and constantly focused at, CBA is provided with sufficiently accurate visual state information. In order to fill the contents of the cases' solution parts, however, CBA must apply inverse dynamics of the soccer simulation. If CBA, for example, observes that the velocity vector of the ball has been changed at time $t+1$ as in the bottom right part of Figure 1, then it can conclude that OPP has executed a $kick(50, -90°)$ action at time t and can use that information to complete the case it created at time step t.

With ongoing observation of dribbling opponent players, CBA's case base $\mathcal{C}$ grows and becomes more and more competent. Therefore, after $|\mathcal{C}|$ exceeds some threshold, CBA can utilize its case base and query it to find a prediction of the action that OPP is going to take in the current time step.

Retrieval and Similarity Measures We model the problem similarity using the local-global principle [2] with identical local similarity measures for all problem attributes, $sim_i(q_i, c_i) = (\frac{q_i - c_i}{max_i - min_i})^2$, where min_i and max_i denote the minimum and maximum value of the domain of the ith feature. The global similarity is formed as a weighted average according to

$$Sim(q,c) = \frac{\sum_{i=1}^{n} w_i \cdot sim_i(q_i, c_i)}{\sum_{i=1}^{n} w_i}$$

where attributes pos_{bnx} and pos_{bny} are weighted twice as much as vel_{pnx} and vel_{pny}.

We perform standard k-nearest neighbor retrieval using a value of $k = 3$ in our experiments. When predicting the class of the solution, i.e. the type of the low-level action (dash or kick), we apply a majority voting, and for the prediction of the action parameters (x and α) we calculate the average over all cases among the k nearest neighbors whose class label matches the majority class.

4 Experimental Results

To evaluate our approach we selected a set of contemporary soccer simulation team binaries (top teams from recent years) and made one of their agents (OPP) dribble for up to 2000 simulated time steps[4]. Our case-based agent CBA was allowed meanwhile to observe OPP and build up its case base. We evaluated CBA's performance in predicting OPP's low-level actions for increasing case base sizes.

Figure 2 visualizes the learning progress against an opponent agent from team WrightEagle. As can be seen, compelling accuracies can be achieved for both, the correctness of the type of the action (dash or kick) as well as for the belonging action parameters. Interestingly, even the relative power / angle of kicks can be predicted quite reliably with a remaining absolute error of less than ten percent / ten degrees.

---

[4] Duration of a regular match is 6000 time steps.

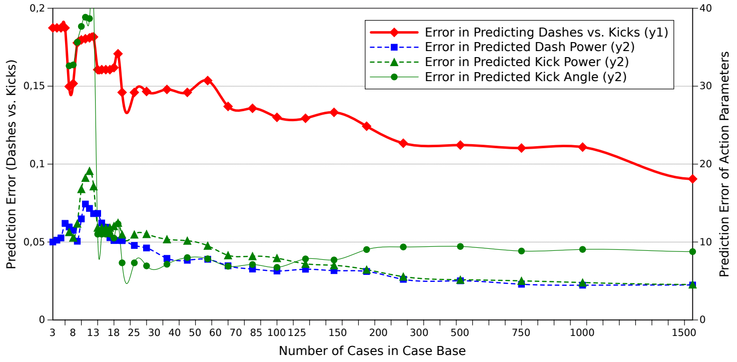


Fig. 2. Progress of CBA's competence in predicting the next low-level actions of a dribbling opponent agent from team WrightEagle. A case base of about 1500 cases was created during the course of 2000 simulated time steps.

Figure 3 focuses on different opponent agents and highlights the fact that a substantial improvement in action type prediction accuracy can be obtained with as little as 100 collected cases. Baseline to all these classification experiments is the error of the "trivial" classifier (black) that predicts each action type to be of the majority class. The right part of Figure 3 presents the recall of both, dash and kick actions. Apparently, dashes are somewhat easier to predict than kicks where, however, the recall of the latter is still above 65% for each of the opponent agents considered.

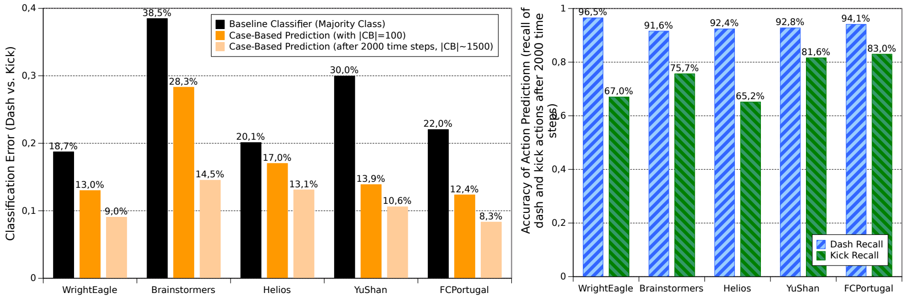


Fig. 3. Left: Case-based prediction of the type (dash or kick) of the next low-level action for opponents from different teams. Right: Recall, i.e. share of dashes that were correctly predicted as dashes and kicks that were correctly predicted as kicks.

In Figure 4, we present aggregate numbers (averages over all opponents) that emphasize how accurately the parameters of an action were predicted, given that the type of the action could be identified correctly. To this end, dash angles α are disregarded since more than 99.2% of all dash actions performed used

$\alpha = 0$, i.e. yielded a dash forward. Here, we compare (a) an “early” case base with only 10 cases, (b) an intermediate one[5] with $|\mathcal{C}| = 100$ as well as (c) one that has resulted from 2000 simulated time steps and contains circa 1500 cases. Interestingly, even in (a) comparatively low errors can be obtained. In (b) and (c), however, the resulting average absolute prediction errors become really competitive (± 2.9 for dash powers x with $x \in [0, 100]$, ± 6.3 for kick powers x with $x \in [0, 100]$, and $\pm 19.7°$ for kick angles α with $\alpha \in [0°, 360°]$).

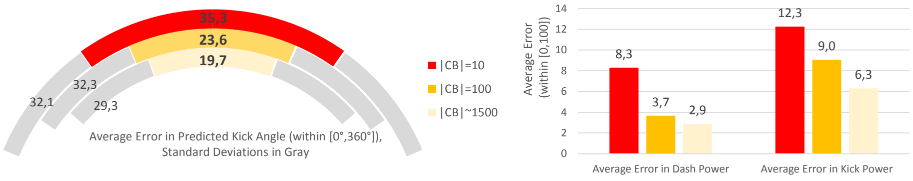


Fig. 4. Exactness of the prediction of the action parameters for different stages during ongoing learning (10, 100, and ≈1500 cases in the case base). Left: Average error of the predicted angle of a kick action. Right: Average error of the predicted relative power of a kick action and dash action (averages over agents from all opponent teams considered).

5 Discussion and Conclusion

Clearly, dribbling opponents are very likely to behave differently when they are disturbed, tackled, or attacked by a nearby opponent. Therefore, the approach presented needs to be extended to “duelling situations” as they frequently arise in real matches. For example, in scenarios like that the dribbler will presumably not just dribble straight ahead, but also frequently execute turn actions (e.g. in order to dribble around its disturber). This represents an aggravation of the action type prediction problem since then three instead of two classes of actions must be considered (dask, kick, turn).

While the case study presented focused solely on non-attacked dribbling opponents, this approach can easily be transferred to related or similar situations where knowing the opponent’s next move is crucial, too. This includes, but is not limited to the behavior of an opponent striker when trying to perform a shoot onto the goal (which typically requires a couple of time steps), the behavior of the shooter as well as the goal keeper during penalty shoot-outs, or the positioning behavior of the opponent goalie (anticipating which can be essential for the striker).

As a next step, we plan to combine the presented case-based prediction of low-level actions with the reinforcement learning-based acquisition of agent behaviors as outlined in Section 2.3. This involves, first, solving the aggravated problem of

---

[5] A case base of a size of about 100 to 500 cases can easily be created within the first half of a match for most players.

correctly recognizing three different classes of low-level actions mentioned at the beginning of this section and, second, a proper utilization of the thereby obtained improved model when learning a higher-level duelling skill using RL. Another interesting direction for future work is the idea to let CBA start off with some opponent model in form of a case-base acquired offline (against, for example, an older version of the team to be faced) and, using appropriate techniques for case base maintenance, to successively replace old experience by new experience gained online during the current match.

References

1. Ahmadi, M., Keighobadi-Lamjiri, A., Nevisi, M., Habibi, J., Badie, K.: Using a Two-Layered Case-Based Reasoning for Prediction in Soccer Coach. In: Proceedings of the International Conference of Machine Learning; Models, Technologies and Applications (MLMTA'03). pp. 181–185. CSREA Press (2003)
2. Bergmann, R., Richter, M., Schmitt, S., Stahl, A., Vollrath, I.: Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning. In: Proceedings of the 9th German Workshop on Case-Based Reasoning. pp. 264–274 (2001)
3. Carvalho, A., Cheriton, D.: Reinforcement Learning for the Soccer Dribbling Task. In: Proceedings of IEEE Conference on Computational Intelligence and Games (CIG). pp. 95–101. Seoul, South Korea (2011)
4. Denzinger, J., Hamdan, J.: Improving Modeling of Other Agents Using Stereotypes and Compactification of Observations. In: Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 1414–1415. New York, USA (2004)
5. Floyd, M., Esfandiari, B., Lam, K.: A Case-Based Reasoning Approach to Imitating RoboCup Players. In: Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference. pp. 251–256. Coconut Grove, USA (2008)
6. Gabel, T., Riedmiller, M.: CBR for State Value Function Approximation in Reinforcement Learning. In: Proceedings of the 6th International Conference on Case-Based Reasoning (ICCBR 2005). pp. 206–221. Springer, Chicago, USA (2005)
7. Gabel, T., Riedmiller, M., Trost, F.: A Case Study on Improving Defense Behavior in Soccer Simulation 2D: The NeuroHassle Approach. In: L. Iocchi, H. Matsubara, A. Weitzenfeld, C. Zhou, editors, RoboCup 2008: Robot Soccer World Cup XII, LNCS. pp. 61–72. Springer, Suzhou, China (2008)
8. Kalyanakrishnan, S., Liu, Y., Stone, P.: Half Field Offense in RoboCup Soccer: A Multiagent Reinforcement Learning Case Study. In: RoboCup-2006: Robot Soccer World Cup X. pp. 72–85. Springer Verlag, Berlin (2007)
9. Kuhlmann, G., Stone, P.: Progress in Learning 3 vs. 2 Keepaway. In: RoboCup-2003: Robot Soccer World Cup VII. pp. 694–702. Springer, Berlin (2004)
10. Noda, I., Matsubara, H., Hiraki, K., Frank, I.: Soccer Server: A Tool for Research on Multi-Agent Systems. Applied Artificial Intelligence 12(2-3), 233–250 (1998)
11. Steffens, T.: Similarity-Based Opponent Modelling Using Imperfect Domain Theories. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG05)
12. Veloso, M., Balch, T., Stone, P.: RoboCup 2001: The Fifth Robotic Soccer World Championships. AI Magazine 1(23), 55–68 (2002)
13. Wendler, J., Bach, J.: Recognizing and Predicting Agent Behavior with Case-Based Reasoning. In: D. Polani and A. Bonarini and B. Browning (editors), RoboCup 2003: Robot Soccer World Cup VII. pp. 729–728. Padova, Italy (2004)

Case-based Local and Global Percept Processing for Rebel Agents

Alexandra Coman[1], Kellen Gillespie [2], Héctor Muñoz-Avila [3]

[1] Department of Electrical and Computer Engineering and Computer Science,
Ohio Northern University, Ada, OH 45810
[2] Knexus Research Corporation, Springfield, VA 22314
[3] Department of Computer Science and Engineering, 19 Memorial Drive West,
Lehigh University, Bethlehem, PA 18015
a-coman@onu.edu, kellen.gillespie@knexusresearch.com, hem4@lehigh.edu

Abstract. Rebel Agents are goal-reasoning agents capable of "refusing" a user-given goal, plan, or subplan that conflicts with the agent's own internal motivation. Rebel Agents are intended to enhance character believability, a key aspect of creating engaging narratives in any medium, among other possible uses. We propose to implement and expand upon a Rebel Agent prototype in eBotworks, a cognitive agent framework and simulation platform. To do so, we will make use of (1) a case-based reasoning approach to motivation-discrepancy-perception, and (2) user input for creating the agents' "emotional baggage" potentially sparking "rebellion".

Keywords: rebel agents, character believability, local and global perceptual processing

1 Introduction

Rebel Agents [6] are motivated, goal-reasoning agents capable of "refusing" a goal, plan, or subplan assigned by a human user or by another agent. This rejection is the result of a conflict arising between the given goal or plan and the agent's own internal motivation. In our previous work, we made the assumption that this motivation is modeled for the purpose of creating character believability [1], a key aspect of engaging narratives in any medium. However, different motivation models are also applicable. In the context of rebel agents, the term "motivation discrepancies" refers to incongruities between a character's motivation and the character's assigned goal and/or course of action. When a motivation discrepancy occurs, depending on the perceived intensity of the incongruity, the Rebel Agent may generate a new goal that safeguards its motivations. While so far explored in the context of interactive storytelling and character believability, the potential applications of rebel agents are by no means limited to this. Such agents can also be useful, for example, in mixed-initiative situations in which the Rebel Agent may have access to information unavailable to its human collaborator, and use this information to decide when to reject a command received from the collaborator.

We are in the process of developing a conceptual framework for Rebel Agents and implementing a Rebel Agent prototype in eBotworks, a cognitive agent framework and simulation platform [15].

In previous work [7], we explained that, for the purpose of detecting and reacting to "motivation discrepancies", eBotworks agents should be made able to perceive and interpret their surroundings in "subjective" ways potentially eliciting "emotion" intense enough to cause rebellion. We showed how eBotworks agent perception, which is by default omniscient and objective, needs to be modified to more closely mimic (or appear to mimic) human perception. We also described that this can be achieved using sensory filters informed by mechanisms of human perception. These mechanisms include gradual perception differentiation, local and global percept processing and, perhaps most importantly for our purposes, the bidirectional connection between perception and emotion. That is, perception can elicit emotion and is, in turn, affected by emotion.

While relying on psychology literature to build these filters, we are ultimately aiming for agents with believable observable behavior, but not based on complex models of cognition.

We aim to endow our prototype Rebel Agent with motivation based on emotionally-charged autobiographical memories. For example, a bot that reaches a location at which something "traumatic" happened in the past might undergo a goal change appropriate to the context. The retrieval of autobiographical memories is to initially occur based on location ecphory [14], that is, location-specific memory cues. They use exact physical locations (i.e. map coordinates) as memory cues. This choice is preferable from a practical standpoint, but does not accurately reflect the way location ecphory works in humans. The characteristics of a location that awaken memories and incite emotion tend to be the sights, sounds, smells, tastes, and tactile sensations pertaining to it, not necessarily its map coordinates. However, while location coordinates are easy to retrieve and to compare, the same cannot be said about complex combinations of percepts.

In previous work [7], we explained how the perception mechanisms of eBotworks can be modified in order to acquire percepts in a more "human-like" manner.

Herein, we approach the challenge of retrieving past percepts and comparing them to current ones using the case-based reasoning model, which is a natural match for this retrieval process. Case-based reasoning literature offers examples of complex case structures and associated similarity measures (e.g., [4][13][18]), allowing us to store and compare complex scene representations, thus taking location ecphory beyond mere map coordinates.

In building a case base consisting of "memories" of percepts and associated emotions, one of the challenges is providing the basis upon which the agents associate emotions to percepts. While this could be accomplished by building a complex inner model of the agent, herein, we discuss a knowledge-engineering-light alternative. This new approach could leverage the chat-based interface of eBotworks, through which users can give agents commands. In our context, human users would be directing the agent how to "feel" in certain situations. That way, the agent, instead of being provided with a complex program dictating how it should behave in various contexts, picks up an "emotional baggage" derived from that human user's personality (or just a "role" that the human user chooses to play). By getting input from different human users, we can produce a range of bots roughly exemplifying various personalities.

Our two contributions herein are:

(1) Exploring a case-based reasoning context for motivation-discrepancy-perception in eBotworks Rebel Agents.
(2) Proposing the use of chat-based user input for creating the agents' "emotional baggage", potentially sparking "rebellion".

Finally, it must be mentioned that, although we approach them in this specific context, local and global percept processing are applicable not only to Rebel Agents, but to any intelligent agents endowed with perception capabilities.

2 Local and Global Percept-Processing and Emotion

Gradual perception differentiation and local and global percept processing have been shown, in psychology literature, to characterize human perception. Human perception has also been shown to stand in bidirectional connection with emotion; percepts of various types can elicit emotional responses [5], while perception can be influenced by emotion and motivation as explained below [9][12][22].

Perception differentiation deals with the steps of the gradual formation of a percept.

Global-first percept processing begins with global features, with local ones becoming increasingly clear in later stages. It has been argued to be induced by positive emotions, such as happiness. Citing [17] and [20], Navon [16] sees perceptual differentiation as always "*proceeding from global structuring towards more and more fine-grained analysis*". As to what makes a feature global, rather than local, Navon describes a visual scene as a hierarchical network, each node of which corresponds to a subscene. Global scenes are higher up in the hierarchy than local ones, and can be decomposed into local ones. More recently, it seems to be agreed upon that, while a widespread tendency towards global-first processing is observed, it cannot be established as a general rule applying to all individuals at all times [22].

Local-first percept processing begins from or focuses on local features. It has been argued to be more likely when under the influence of negative emotions, such as stress and sadness. However, strong motivation has also been shown to be capable of inducing local-first processing [11]. Individuals with certain personality disorders have been hypothesized to be inclined towards local precedence. Yovel, Revelle, and Mineka [21] state that obsessive-compulsive personality disorder has been connected to *"excessive visual attention to small details"*, as well as "*local interference*": an excessive focus on small details interfering with the processing of global information. The same preference for local processing has been associated with autism spectrum disorders [10].

The tendency towards global or local processing has also been theorized to be culture-specific: certain cultures have been shown to favor local precedence [8].

Connections between perception, emotion, and motivation are discussed at length by Zadra and Clore [22]. Their survey covers the effects of emotion and mood on global vs. local perception, attention, and spatial perception.

3 Local and Global Percept Processing for Rebel Agents in eBotWorks

eBotworks [15] is a software platform for designing and evaluating communicative autonomous systems in simulated environments. “Communicative” autonomous systems are those that can interact with the environment, humans, and other agents in robust and meaningful ways, including the use of natural language. eBotworks tasks have so far been limited to path-finding and obstacle-avoidance-type tasks (Figure 1), and have not been concerned with character believability.

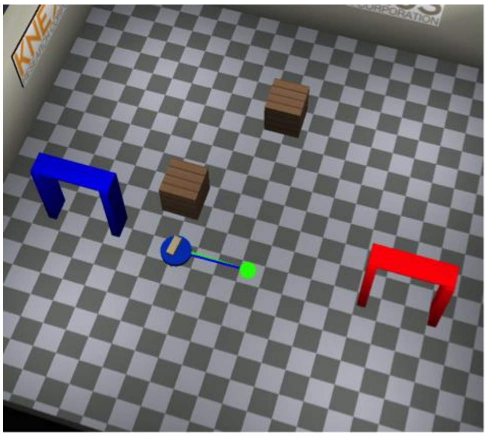


Figure 1. An eBotworks scene with an eBot performing obstacle avoidance.

In previous work [7], we designed scenarios showcasing emotion-influenced perception for possible future implementation in eBotworks. We then discussed how the implementation of these scenarios might be achieved with existing components of the framework.

We will first reiterate these scenarios before we explain how the newly-proposed mechanisms can be used to achieve them. The scenarios are based on the following assumptions: (1) the agent is a Rebel Agent [6] endowed with an autobiographical memory model in which memories are connected to emotions, (2) default perception is global-first, (3) agents have current “moods” (emotional states) which can be neutral, positive or negative, with the “neutral” mood being the default one, (4) moods can change as a result of perceiving scenes evoking autobiographical memories with emotional associations, (5) mood affects perception in the ways described in Section 2, (6) all scenarios take place on the same map, (7) in all scenarios, the agent has been assigned a goal that involves movement to a target location on the map; based on its reaction to scenes perceived on its way to the target, the agent may or may not rebel; when a rebellion threshold is reached, the agent does rebel, (8) in all scenarios, the agent perceives two scenes on its way to the target; the perception of the first scene may or may not affect the agent's current mood, which, in turn, may influence how the second scene is perceived.

- *Scenario 1*: On the way to its target location, the agent perceives a box. This evokes no emotions, as there are no connections to the box in the autobiographical memory of the agent. Then, the agent perceives the second scene: a traffic-cone-lined driving course, using global-precedence perception. The agent's emotion changes to a slightly-positive one, as it "enjoys" driving through traffic-cone-lined driving courses. This does not elicit a goal change.
- *Scenario 2*: On the way to its target location, the agent perceives a box. In the agent's autobiographical memory, the box has positive emotional associations. This changes the agent's mood to a positive one. Positive moods favor global perception, so they do not change the agent's default perception type. The agent perceives the traffic-cone-lined driving course using global-precedence perception. The agent's mood remains positive. This does not elicit a goal change.
- *Scenario 3*: On the way to its target location, the agent perceives a box. In the agent's autobiographical memory, the box has negative emotional associations. Therefore, the agent's current mood changes to a negative one. Soon afterwards, the agent perceives the traffic-cone-lined driving course. Due to the agent's mood, local interference occurs, and the agent largely ignores the overall scene, while focusing on the color of the cones (which is similar to that of the box), which reminds it of a sad occurrence from the past, like a collision. This changes the agent's mood to a more intensely negative one, which causes the rebellion threshold to be reached and the agent to "rebel".

4 Case-Based Reasoning for Location Ecphory

Ecphory is the remembrance, caused by a memory trigger, of a past event. In the case of location ecphory, this trigger is a location with which the memory is associated.

Gomes, Martinho, and Paiva [14] use map coordinates as location-ecphory triggers. While this is easier from a practicality standpoint, the authors admit it does not accurately reflect the way location ecphory works in humans. Location coordinates (unless physically perceived, with some emotional associations) are unlikely to awaken memories and incite strong emotion. Instead, it is the sights, sounds, smells, tastes, and tactile sensations pertaining to a place that work to achieve this recollection. Thus, if these traits change beyond recognition, the location's function as a memory cue is invalidated.

Retrieving stored memories is a natural match for the case-based reasoning model, which was inspired by the psychological mechanisms underlying memory storage and recollection. Furthermore, case-based reasoning literature contains ample coverage of similarity measures between complex case structures that are not trivially comparable, including graphs, plans, and case structures based on object orientation, which is precisely what we need for implementing our Rebel Agent prototype in eBotworks. By using case-based reasoning similarity measures, we intend to expand location ecphory beyond just map coordinates.

4.1 Case Structure and Similarity Measures

To achieve the emotional location-ecphory effect we are aiming for, each case should contain two essential pieces of information: (1) a past percept, and (2) an emotional reaction associated with that percept.

The ways in which we model these pieces of information can vary in complexity. The percept can be a complex scene or a very specific subscene, such as an individual object or something slightly more general such as a set of objects on a table. The emotional reaction can consist of a simple, basic emotion (e.g. "joy") or of a complex, layered conglomerate of emotions, each experienced at a different degree of intensity.

Due to the characteristics of our simulation platform, we are, for now, focusing on visual ecphory triggers, although triggers of a different nature (e.g. gustatory and olfactory) certainly function in the real world.

In choosing our case structure, we are influenced by the description that Navon [16] gives of a visual scene as a hierarchical network, each node of which corresponds to a subscene. Global scenes are higher up in the hierarchy than local ones, and can be decomposed into local ones. Global-first processing proceeds from global scenes, local-first processing from local ones. We do not, however, aim at matching any psychological model of perception differentiation perfectly through our case representation.

To approximate this hierarchical structure, we propose a model inspired by object-oriented ([3][2] - Section 4.4) and graph-based ([19][2] - Section 4.5) case structures.

A scene hierarchy is not equivalent to a class inheritance hierarchy, though there are clear similarities between the two. The reason is that in a class hierarchy, classes lower down in the hierarchy incorporate the attributes of classes higher up, whereas in the scene/subscene hierarchy, the inverse takes place: the root scene incorporates information from all lower nodes, because the complete scene is composed of all subscenes.

It is to be noted that the rather simple description above does not accurately capture human perception, in which a global scene is perceived as a general outline with vague details that become clear while travelling downwards in the hierarchy. Therefore, the details in the lower nodes are then incorporated (potentially completely) into the higher nodes. If perception proceeds in a global-first manner and is not prolonged, these lower levels may not be reached.

The similarity methods of Bergmann and Stahl [3] allow objects at different levels in the class hierarchy to be compared. This is especially useful, as we have no guarantees that two subscenes we are comparing are at similar hierarchical levels.

However, our situation is even more challenging: not only are the scenes that we are comparing different and at different hierarchical levels, but even their respective hierarchies can be different and correspond to varied scenes (unless the scenes that can be perceived are highly controlled and limited). Despite this challenge, we believe that the local and global similarity measures proposed by Bergmann and Stahl [3] can be adapted to be used for local and global perception, respectively. The perception setting of the agent at a given time (e.g. global after perceiving the box in Scenarios 1 and 2; local after perceiving the box in Scenario 3) will determine where in the hierarchy we look for the similarity.

For simplification, we can assume that the cases are collections of objects, and do not take into account the spatial positioning of the objects in a scene in relation to one another.

4.2 Populating the Case Base

In order to populate a case base consisting of "memories" of percepts and associated emotions, we must first provide a mechanism allowing agents' percepts to be associated with emotions.

Truly human-like agents would be able to generate emotions themselves. This would be partially based on (1) the personality with which the agent would have been endowed (which could dictate, for example, that the agent is not easily frightened), and (2) general rules about ways in which people tend to react to certain situations (e.g. a gruesome scene tends to cause shock). Thus, making agents able to generate emotions in response to percepts would require providing them with at least one of these two models.

We are interested in exploring a knowledge-light alternative to this challenge. This approach can leverage the chat interface of eBotworks (or alternative eBotworks mechanisms) and is based on the idea of having human users direct the agent on how to "feel" in certain situations. Thus, the agent acquires an "emotional baggage" derived from that human user's personality or a "role" that the human user chooses to play. Some bots, for instance, could be directed to be more "impressionable" than others.

Let us re-examine Scenario 3, where the agent perceives a box with negative emotional associations. With this approach, this association would not exist because the bot previously got hurt in the vicinity of the box, but rather because the bot was previously told that the box should make it "feel sad".

While we only propose this mechanism for the purpose of attaching specific emotions to scenes, it could later be applied more broadly within the context of motivation discrepancies and Rebel Agents. For instance, it could also be used to assign meaning to scenes, so that the agent can match scenes similar in meaning (e.g. "a quarrel") rather than just in their constitutive elements. With this ability, agents can then match emotion to meaning (e.g. witnessing a quarrel causes stress), rather than just to specific scenes and subscenes.

Currently, the chat interface of eBotworks is used to issue commands to agents in a simulated environment. For example, a user can enter "Go here" and click on a location on the current map; if the command is successful, the agent reacts by moving to the specified location.

To explain how this system could be used for our purposes, let us first assume that the bot is facing a scene containing a box. One option would be for the user to simply say one of several words corresponding to several emotions "understood" by the system, e.g. "sad". In this case, the agent would take a "snapshot" of the scene it is facing and store it together with the associated emotion, sadness.

However, memories of strong emotions can be associated with very specific subscenes, rather than to an entire complex scene (e.g. excitement associated with a logo on the envelope containing a college acceptance letter). Moreover, the subscene that attention ends up focusing on in such situations is not necessarily related to the emotion itself. Instead, it could contain items that just happen to be there when the emotionally charged event occurs (e.g., a cup that happens to be on a nearby table while a severe argument is taking place).

To handle this possibility, we can allow the user to specify an object in the scene to which to associate the emotion by clicking on the object first, then saying the word corresponding to the emotion. In Scenario 3, clicking on a box then saying “sad” can cause the agent to switch to a sad mood and experience local interference in perception. Another necessary addition to typical eBotworks usage will be to have the agent convey, through console messages, (and, later, possibly, through visual representations on the map) what objects it is currently focusing on and what moods it is experiencing. This will enhance believability by providing insight into the agent’s motivations and into the emotional justification behind its actions.

5 Conclusions

We have discussed using the case-based model for the purpose of creating location-ecphory-based motivation-discrepancy mechanisms for Rebel Agents, addressing the challenge of retrieving emotionally-charged past percepts and comparing them to current ones.

Our two main contributions herein are:

(1) Exploring a case-based reasoning context for motivation-discrepancy-perception in eBotworks Rebel Agents.
(2) Proposing the use of chat-based user input for creating the agents’ “emotional baggage”, potentially sparking “rebellion”.

Acknowledgements. We thank the reviewers for their comments and suggestions, some of which have been integrated into the final version of the paper, others of which will be of help to us in our future work.

References

1. Bates, J. (1994). The role of emotion in believable agents. *Communications of the ACM*, 37(7), 122-125.
2. Bergmann, R. (2002). Experience management: foundations, development methodology, and Internet-based applications. Springer-Verlag.
3. Bergmann, R., & Stahl, A. (1998). Similarity measures for object-oriented case representations (pp. 25-36). Springer Berlin Heidelberg.
4. Börner, K. (1994). Structural similarity as guidance in case-based design. In Topics in case-based reasoning (pp. 197-208). Springer Berlin Heidelberg.
5. Clore G.L., & Ortony A. (2008) Appraisal theories: How cognition shapes affect into emotion. In *Handbook of Emotion 3,* 628–642 New York: Guilford Press..

6. Coman, A., & Muñoz-Avila, H. (2014). Motivation discrepancies for rebel agents: Towards a framework for case-based goal-driven autonomy for character believability. *Proceedings of the 22nd International Conference on Case-Based Reasoning (ICCBR) Workshop on Case-based Agents*.
7. Coman, A, Gillespie, K., and Muñoz-Avila, H. (2015). Believable Emotion-Influenced Perception: The Path to Motivated Rebel Agents, *Proceedings of the Advances in Cognitive Systems (ACS) Workshop on Goal Reasoning*, Atlanta, Georgia, May 28th - 31st.
8. Davidoff, J., Fonteneau, E., & Fagot, J. (2008) Local and global processing: Observations from a remote culture. *Cognition*;108(3):702–709.
9. Easterbrook J. (1959) The effect of emotion on cue utilization and the organization of behavior. *Psychol. Rev.*, 66(3):183–201.
10. Frith, U. (1989) *Autism: Explaining the enigma*. Oxford, UK: Blackwell Scientific Publications.
11. Gable P.A., & Harmon-Jones E. (2008) Approach-motivated positive affect reduces breadth of attention. *Psychol. Sci*., 19:476–482.
12. Gasper K, & Clore G.L. (2002) Mood and global versus local processing of visual information. *Psychol. Sci.*, 13:34–40
13. Goel, A. K., & Craw, S. (2005). Design, innovation and case-based reasoning. The Knowledge Engineering Review, 20(03), 271-276.
14. Gomes, P. F., Martinho, C., & Paiva, A. (2011). I've Been Here Before! Location and Appraisal in Memory Retrieval. *Int. Conf. on Autonomous Agents and Multiagent Systems*.
15. Gupta K. M., & Gillespie K. (2015) eBotworks: A software platform for developing and evaluating communicative autonomous systems. AUVSI Unmanned Systems, Atlanta, GA.
16. Navon, D. (1977). Forest before trees: The precedence of global features in visual perception. *Cognitive Psychology*. 9(3):, 353–383.
17. Palmer, S.E. (1975) Visual perception and world knowledge: Notes on a model of sensory-cognitive interaction. In D. A. Norman , D. E. Rumelhart, and the LNR Research Group, *Explorations in cognition*, San Francisco, CA: Freeman.
18. Smyth, Barry, and Padraig Cunningham. (1992) Déjà Vu: A Hierarchical Case-Based Reasoning System for Software Design. ECAI. Vol. 92.
19. Vattam, S., Aha, D.W., & Floyd, M. (2014). Case-based plan recognition using action sequence graphs. In Proceedings of the Twenty-Second International Conference on Case-Based Reasoning, (pp. 495-510). Cork, Ireland: Springer.
20. Winston, P.H. (1973) Learning to identify toy block structures. In R.L. Solso (Ed.) *Contemporary issues in cognitive psychology: The Loyola Symposium*, Washington D.C.

21. Yovel I., Revelle W., & Mineka S. (2005). Who sees trees before forest? The obsessive compulsive style of visual attention. *Psychological Science*, 16, 123-129.
22. Zadra J.R., & Clore G.L. (2011) Emotion and perception: The role of affective information. *Wiley Interdisciplinary Reviews: Cognitive Science*.

Predicting the Outcome of Small Battles in StarCraft *

Antonio A. Sánchez-Ruiz

Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid (Spain)
`antsanch@fdi.ucm.es`

Abstract. Real-Time Strategy (RTS) games are popular testbeds for AI researchers. In this paper we compare different machine learning algorithms to predict the outcome of small battles of marines in StarCraft, a popular RTS game. The predictions are made from the perspective of an external observer of the game and they are based only on the actions that the different units perform in the battlefield. Our empirical results show that case-based approaches based on k-Nearest Neighbor classification outperform other standard classification algorithms like Linear and Quadratic Discriminant Analysis or Support Vector Machines.

Keywords: Prediction, StarCraft, Linear and Quadratic Discriminant Analysis, Support Vector Machines, k-Nearest Neighbors

1 Introduction

Real-Time Strategy (RTS) games are popular testbeds for AI researchers [4] because they provide complex and controlled environments in which to carry out different experiments. In this paper we assume the role of an external observer of the game that tries to predict the outcome when the armies of two different players engage in combat. As a spectator of the game, we can only base the predictions on the actions of the different units in the battlefield. From this perspective, we can consider each army as a group of agents working in a coordinated manner to defeat the other army. We know that the units in the game are not really agents because they are not autonomous (in fact they are controlled by a human player or by the internal AI of the game), but from the perspective of an external observer we only see several units performing actions in a simulation, and we do not know whether those actions are consequence of individual decisions or some superior intelligence. Therefore, our approach to prediction in RTS games could be applied as well to multi-agent simulations.

The ability to predict the outcome of battles is interesting because it can be used to dynamically modify the strategy of the player. For example, the player could decide to change the composition of the army, to bring more troops into

---

* Supported by Spanish Ministry of Economy and Competitiveness under grant TIN2014-55006-R

In Proceedings of the ICCBR 2015 Workshops. Frankfurt, Germany.

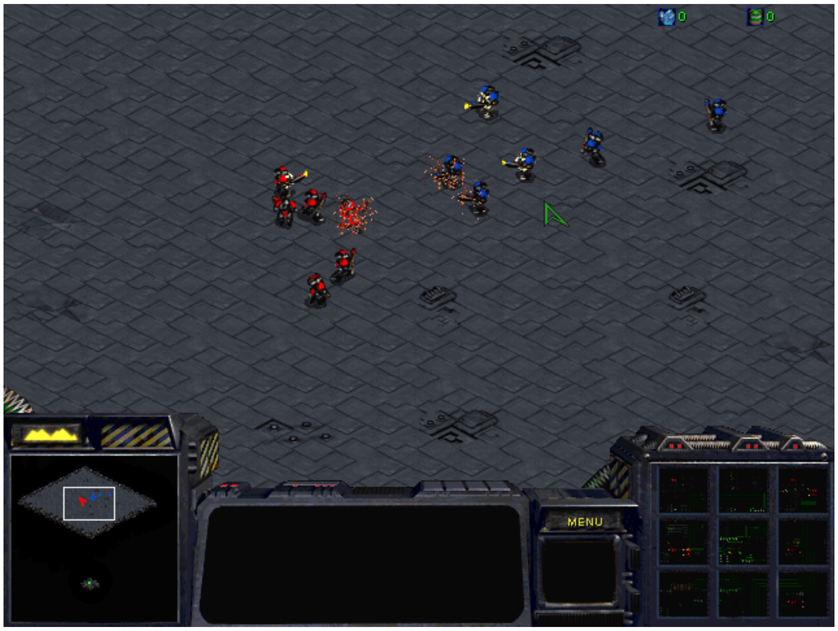


Fig. 1: Screenshot of our custom map: Battle of Marines

the battlefield or deploy them differently, or even to flee if the prospects are not good. In general, an agent able to make predictions (running an internal simulation based on what he knows) might be able to adapt his behavior more successfully than other agent without this ability.

In this work, we compare classical classification algorithms like Linear and Quadratic Discriminant Analysis, Support Vector Machines, and two instance-based classifiers based on the retrieval of the k-Nearest Neighbors (kNN). kNN classifiers can be seen as simple Case-based Reasoning (CBR) systems that only implement the retrieval phase of the CBR cycle. In this paper we study the accuracy of the prediction during the course of the battle, the number of games that each algorithm needs to learn, and the stability of the prediction over time.

The rest of the paper is organized as follows. Sections 2 and 3 describe the scenario used in the experiments, the process to extract the data for the analysis and the features chosen to represent the game state. Sections 4 and 5 explain the different classification algorithms and the results obtained. The paper concludes with the related work, and some conclusions and directions for future research.

2 Battles of Marines in StarCraft

StarCraft[1] is a popular RTS game in which players have to harvest resources, develop technology and build armies combining different types of units to defeat

---

[1] http://us.blizzard.com/en-us/games/sc/

game	frame	units1	life1	area1	units2	life2	area2	distance	winner
1	0	6	40	2772	6	40	2520	1309.903	1
1	3	6	40	2772	6	40	2520	1309.903	1
1	6	6	40	2736	6	40	2925	1302.857	1
1	9	6	40	2964	6	40	2923	1282.317	1
1	12	6	40	3876	6	40	2905	1266.277	1
1	15	6	40	4332	6	40	3045	1246.241	1

Table 1: Examples of game states extracted from a Starcraft game trace.

the other players. The combination of different types of units and abilities, and the dynamic nature of the game force players to develop strategies at different levels. At the *macro* level, players have to decide the amount of resources invested in map exploration, harvesting, technology development, troops, offensive and defensive forces, among others. At the *micro* level players have to combine different types of units, locate them in the map and use their abilities. In this paper we focus on small battles, that is, at the micro level.

StarCraft also provides a map editor to create custom games. Using this tool, we have created a simple combat scenario (Figure 1) in which each player controls a small army of 6 *terran marines* (marines are basic ground combat units with ranged attack). The game always begins with the same initial configuration, each army located on opposite sides of the map, and the game ends when all the units of one player are destroyed. In this type of scenario it is very important to strategically locate the units on the map to take advantage of the range attack and concentrate the fire on a few units to destroy them as soon as possible.

3 Data Collection and Feature Selection

In order to obtain the data to train the different classifiers, we played 200 games collecting traces that describe the evolution of the games. We configured the map so the internal game AI controls both players so (1) we can automatically play as many games as required, and (2) we know that all the games are well balanced (since the StarCraft AI is playing against itself). Finally, there is a third player that only observes the game (it does not intervene) and extracts the game traces to a file so they can be analyzed later[2].

The data set contains *traces* of 200 games in which player 1 won 119 times and player 2 the remaining 81. They are very fast games with an average duration of 19.12 seconds. In each trace we store the *game state* 6 times per second, so each game is described with approximately 114 games states or *samples.*

Each game state is stored using a vector of features (Table 1) that represents the combat power of each army and the strategic deployment of the troops in the map. The combat power is represented using the number of units alive in each

---

[2] We use the BWAPI framework to extract information during the game (https://github.com/bwapi/bwapi).

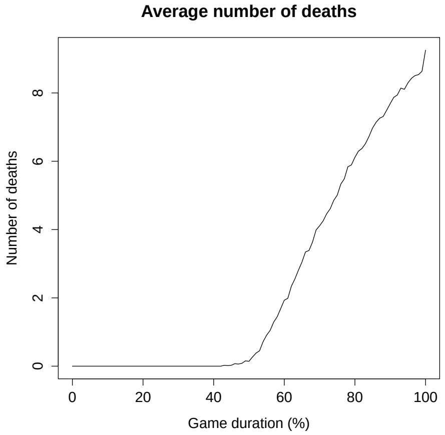


Fig. 2: Average number of deaths during the game.

army and their average life. To represent the strategic distribution of troops in the map we compute the area of the minimum axis aligned rectangle containing the units of each player and the distance between the centers of both rectangles. The rectangle area is a measure of the dispersion of the units, and the distance between the centers indicates how close the two armies are. Each game state is labeled later with the winner of that particular game. The table also shows the game and the current frame for clarity (1 second is 18 game frames although we only sample 6 of them), but we do not use those values in the prediction.

The features to describe the strategic distribution of the troops in the map are especially important during the first seconds of the game. Figure 2 shows the average number of dead units during the 200 games. As we can see, during the first half of the game the armies are approaching each other and the fight does not start until the second half. Thus, during the first seconds of the game the predictions will depend only on the relative location of the units.

4 Classification algorithms

We will compare the following classification algorithms in the experiments:

- Linear Discriminant Analysis (LDA) [8] is classical classification algorithm that uses a linear combination of features to separate the classes. It assumes that the observations within each class are drawn from a Gaussian distribution with a class specific mean vector and a covariance matrix common to all the classes.
- Quadratic Discriminant Analysis (QDA) [9] is quite similar to LDA but it does not assume that the covariance matrix of each of the classes is identical, resulting in a more flexible classifier.

Classifier	Accuracy	Parameters
Base	0.5839	
LDA	0.7297	
QDA	0.7334	
SVM	0.7627	kernel = radial, C = 1, sigma = 0.3089201
KNN	0.8430	k = 5
KKNN	0.9570	kernel = optimal, kmax = 9, distance = 2

Table 2: Classification algorithms, configuration parameters and global accuracy.

- Support Vector Machines (SVM) [7] have grown in popularity since they were developed in the 1990s and they are often considered one of the best *out-of-the-box* classifiers. SVM can efficiently perform non-linear classification using different kernels that implicitly map their inputs into high-dimensional feature spaces. In our experiments we tested 3 different kernels (lineal, polynomial and radial basis) obtaining the best results with the radial basis.
- k-Nearest Neighbour (kNN) [2] is a type of instance-based learning, or lazy learning, where the function to learn is only approximated locally and all computation is deferred until classification. The kNN algorithm is among the simplest of all machine learning algorithms and yet it has shown good results in several different problems. The classification of a sample is performed by looking for the k nearest (in Euclidean distance) training samples and deciding by majority vote.
- Weighted K-Nearest Neighbor (kkNN) [10] is a generalization of kNN that retrieves the nearest training samples according to Minkowski distance and then classifies the new sample based on the maximum of summed kernel densities. Different kernels can be used to weight the neighbors according to their distances (for example, the *rectangular* kernel corresponds to standard un-weighted kNN). We obtained the best results using the *optimal* kernel [14] that uses the asymptotically optimal non-negative weights under some assumptions about the underlying distributions of each class.

The three first algorithms use the training data (labeled game states in our experiments) to infer a generalized model, and then they use that model to classify the test data (new unseen game states). The last two algorithms, however, use the training data as cases and the classification is made based on the most similar stored cases. All the experiments have been run using the R statistical software system [11] and the algorithms implemented in the packages *caret*, *MASS*, *e1071*, *class* and *kknn*.

5 Analyzing the results

Table 2 shows the configuration parameters used in each classifier and its accuracy computed as the ratio of samples (game states) correctly classified. The optimal parameters for each classifier were selected using repeated 10-fold cross

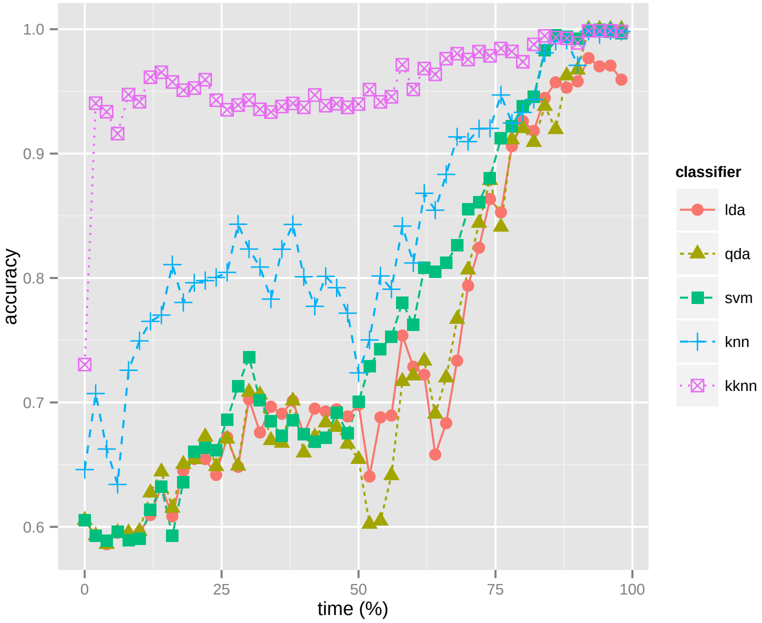


Fig. 3: Accuracy of the classifiers during the game.

validation over a wide set of different configurations. The accuracy value has been calculated as the average of 32 executions using 80% of the samples as the training set and the remaining 20% as the test set.

The *base* classifier predicts the winner based only on the proportion of samples belonging to each class (58.39% of the samples correspond to games won by player 1) and it is useful only as a baseline to compare the other classifiers. LDA, QDA and SVM obtain accuracy values ranging from 72% to 76%. The two instance-based algorithms, on the other hand, obtain higher precision values. It is especially impressive the result of kkNN that is able to predict the winner 95.70% of the times. These results seem to indicate that, in this particular scenario, it is quite difficult to obtain a generalized model, and local based methods perform much better.

The global accuracy value may not be informative enough because it does not discriminate the time of the game when the prediction is made. It is reasonable to expect the accuracy of the predictions to increase as the game evolves as it is shown in Figure 3. The x-axis represents the percentage of elapsed time (so we can uniformly represent games with different duration) and the y-axis the average accuracy of each classifier for game states from that time interval.

Selecting a winner during the second half of the game is relatively easy since we can see how the battle is progressing, but during the first half of the game the prediction problem is much more difficult and interesting since we only see

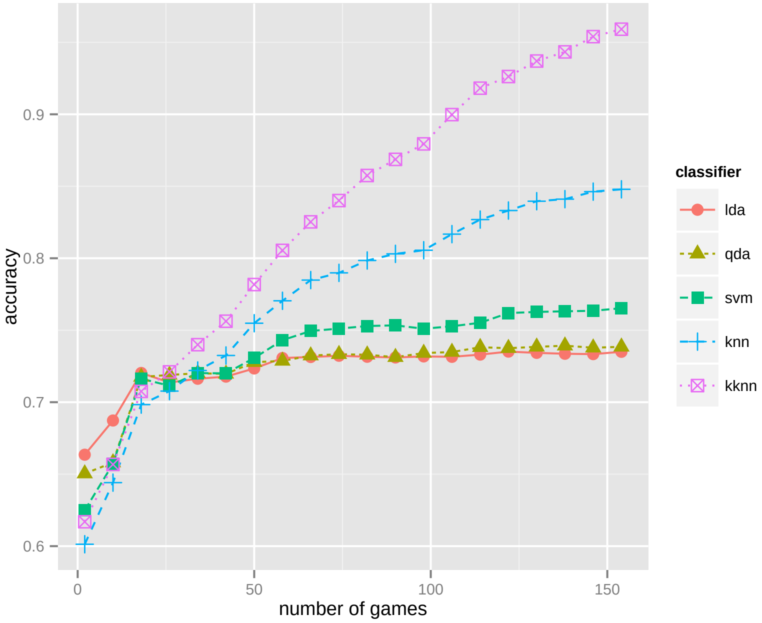


Fig. 4: Accuracy of the classifiers vs. the number of training games.

the formation of the armies (*pre-battle* vs. *during battle* prediction). LDA, QDA and SVM do not reach 90% of accuracy until the last quarter of the game. kNN is able to reach the same accuracy at 66% of the game. The results of kkNN are spectacular again, classifying correctly 90% of the game states from the first seconds. kkNN is the only algorithm able to effectively find useful patters in the training data before the armies begin to fight. Our intuition is that the training data is biased somehow, probably because the StarCraft AI is playing against itself and it does not use so many different attack strategies. In any case, kkNN seems to be the only algorithm to effectively predict the outcome of the battle from the first moves of each army.

Another important aspect when choosing a classifier is the volume of training data they need to perform well. Figure 4 shows the accuracy of each classifier as we increase the number of games used during the training phase. In the first 20 games all the algorithms perform similarly but then only kNN and kkNN keep improving fast. It makes sense for instance-based algorithms to require a large number of samples to achieve their highest degree of accuracy in complex domains, while algorithms that infer general models stabilize earlier but their prediction is more biased.

Finally, Figure 5 shows the stability of the predictions. We divided the game in 20 intervals of 5% of time. The y-axis represents the number of games for which the classifier made a prediction in that time interval that remained stable

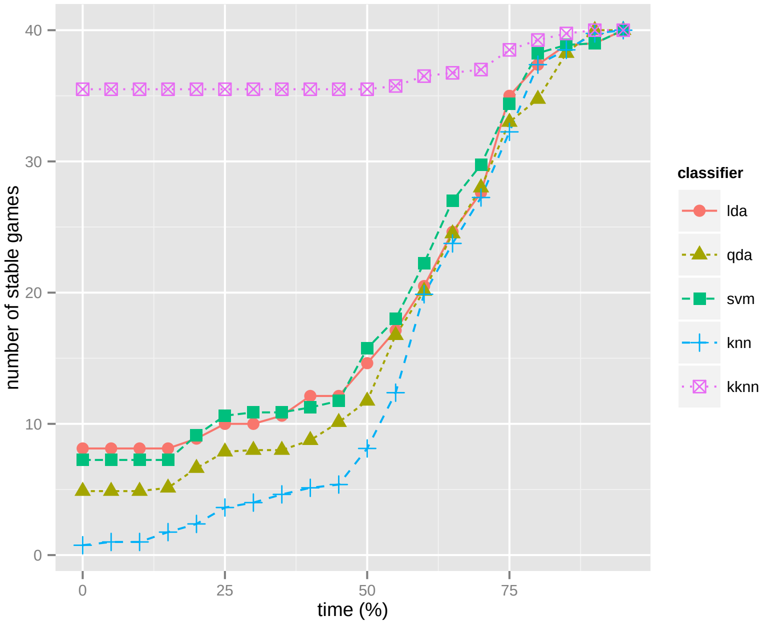


Fig. 5: Number of games for which each classifier becomes stable at a given time.

for the rest of the game. For example, the y value for $x = 0$ represents the number of games in which the prediction became stable during the first time interval (0-4.99% of the game). Most of the classifiers need to wait until the last quarter of the game to be stable in 80% of the games, except kkNN that is very stable from the beginning. There are a few games, however, in which all the classifiers are wrong until the end of the game because the army that was winning made bad decisions during the last seconds.

In conclusion, instance-based classifiers seems to perform better in our scenario, and kkNN in particular is the only algorithm that is able to effectively find useful patters in the training data before the armies begin to fight. It is also the most stable and it only performs worst than the other algorithms where there is a very small number of training games available.

6 Related work

RTS games have captured the attention of AI researchers as testbeds because they represent complex adversarial systems that can be divided into many interesting subproblems [4]. Proof of this are the different international competitions in AIIDE and CIG conferences. We recommend [12] for a complete overview of the existing work on this domain, the specific AI challenges and the solutions that have been explored so far.

There are several papers regarding the combat aspect of RTS games. [6] describes a fast Alpha-Beta search method that can defeat commonly used AI scripts in small combat scenarios. It also presents evidence that commonly used combat scripts are highly exploitable. A later paper [5] proposes new strategies to deal with large StarCraft combat scenarios.

Several different approaches have been used to model opponents in RTS games in order to predict the strategy of the opponents and then be able to respond accordingly: decision trees, kNN, logistic regression [17], case-based reasoning [1, 3], bayesian models [16] and evolutionary learning [13] among others.

A paper very related to our work is [15], where authors present a Bayesian model that can be used to predict the outcome of isolated battles, as well as to predict what units are needed to defeat a given army. Our approach is different in the sense that we try to predict the outcome as the game progresses and our battles begin with 2 balanced armies (same number and type of units). We use tactical information regarding the location of the troops and we use StarCraft to run the experiments and not a battle simulator.

7 Conclusions and Future work

In this paper we compare different machine learning algorithms in order to predict the outcome when two small marine armies engage in combat in the StarCraft game. The predictions are made from the perspective of an external game observer so they are based only on the actions of the individual units. The proposed approach is not limited to RTS games and can be used in other domains like multi-agent simulations, since it does not depend on whether the actions are decided by each unit autonomously or by a global manager. Our results indicate that, in this scenario, instance-based classifiers such as kNN and kkNN behave much better than other classifiers that try to infer a general domain model in terms of accuracy, size of the training set and stability.

There are several possible ways to extend our work. We have only considered a small battle scenario with a limited number of units. As the number and diversity of units increases, the number of possible combat strategies also grows creating more challenging problems. Our map is also quite simple and flat, while most of the StarCraft maps have obstacles, narrow corridors, wide open areas and different heights providing locations with different tactical value. The high accuracy values obtained by kkNN from the first seconds of the battle make us suspicious about the diversity of the strategies in the recorded games. We plan to run new experiments using human players to verify our results. Finally, our predictions are based on static pictures of the current game state. It is reasonable to think that we could improve the accuracy if we consider the evolution of the game and not just the current state to predict the outcome of the battle.

References

1. Aha, D.W., Molineaux, M., Ponsen, M.: Learning to win: Case-based plan selection in a real-time strategy game. In: in Proceedings of the Sixth International

Conference on Case-Based Reasoning. pp. 5–20. Springer (2005)

2. Altman, N.S.: An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. American Statistician 46, 175–185 (1992)
3. Auslander, B., Lee-Urban, S., Hogg, C., Muñoz-Avila, H.: Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. In: Advances in Case-Based Reasoning, 9th European Conference, ECCBR 2008, Trier, Germany, September 1-4, 2008. Proceedings. pp. 59–73 (2008)
4. Buro, M., Furtak, T.M.: RTS games and real-time AI research. In: In Proceedings of the Behavior Representation in Modeling and Simulation Conference (BRIMS. pp. 51–58 (2004)
5. Churchill, D., Buro, M.: Portfolio greedy search and simulation for large-scale combat in starcraft. In: 2013 IEEE Conference on Computational Inteligence in Games (CIG), Niagara Falls, ON, Canada, August 11-13, 2013. pp. 1–8 (2013)
6. Churchill, D., Saffidine, A., Buro, M.: Fast Heuristic Search for RTS Game Combat Scenarios. In: Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE-12, Stanford, California, October 8-12, 2012 (2012)
7. Cortes, C., Vapnik, V.: Support-Vector Networks. Mach. Learn. 20(3), 273–297 (Sep 1995)
8. Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7(7), 179–188 (1936)
9. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics, Springer New York Inc., New York, NY, USA (2001)
10. Hechenbichler, K., Schliep, K.: Weighted k-Nearest-Neighbor Techniques and Ordinal Classification (2004)
11. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning: with Applications in R. Springer Texts in Statistics, Springer (2013)
12. Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., Preuss, M.: A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. IEEE Trans. Comput. Intellig. and AI in Games 5(4), 293–311 (2013)
13. Ponsen, M.J.V., Muñoz-Avila, H., Spronck, P., Aha, D.W.: Automatically Acquiring Domain Knowledge For Adaptive Game AI Using Evolutionary Learning. In: Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA. pp. 1535–1540 (2005)
14. Samworth, R.J.: Optimal weighted nearest neighbour classifiers. Ann. Statist. 40(5), 2733–2763 (10 2012)
15. Stanescu, M., Hernandez, S.P., Erickson, G., Greiner, R., Buro, M.: Predicting Army Combat Outcomes in StarCraft. In: Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE-13, Boston, Massachusetts, USA, October 14-18, 2013 (2013)
16. Synnaeve, G., Bessière, P.: A Bayesian Model for Plan Recognition in RTS Games Applied to StarCraft. In: Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2011, October 10-14, 2011, Stanford, California, USA (2011)
17. Weber, B.G., Mateas, M.: A Data Mining Approach to Strategy Prediction. In: Proceedings of the 5th International Conference on Computational Intelligence and Games. pp. 140–147. CIG'09, IEEE Press, Piscataway, NJ, USA (2009)

Multi-Agent Case-Based Diagnosis in the Aircraft Domain

Pascal Reuss[1][2], Klaus-Dieter Althoff[1][2], Alexander Hundt[1], Wolfram Henkel[3], and Matthias Pfeiffer[3]

[1] German Research Center for Artificial Intelligence
Kaiserslautern, Germany
`http://www.dfki.de`
[2] Institute of Computer Science, Intelligent Information Systems Lab
University of Hildesheim, Hildesheim, Germany
`http://www.uni-hildesheim.de`
[3] Airbus
Kreetslag 10 21129 Hamburg, Germany

Abstract. Aircraft diagnosis is a highly complex topic. Many knowledge sources are required and have to be integrated into a diagnosis system. This paper describes the instantiation of a multi-agent system for case-based aircraft diagnosis based on the SEASALT architecture. This system will extend a existing rule-based diagnosis system, to make use of the experience of occurred faults and their solutions. We describe the agents within our diagnosis system, the planned diagnosis workflow and the current status of the implementation. For the case-based agents, we give an overview of the initial case structures and similarity measures. In addition, we describe some challenges we have during the development of the multi-agent system, especially during the knowledge modeling.

1 Introduction

An aircraft is a complex mechanism, consisting of many subsystems. Occurring faults cannot be easily tracked to their root cause. A fault can be caused by one system, by the interaction of several systems or by the communication line between systems. Finding the root cause can be very time and resource consuming. Therefore the use of experience from successfully found and solved root causes can be very helpful for aircraft diagnosis. This paper describes the instantiation of a multi-agent system (MAS) based on the SEASALT architecture. The MAS contains several Case-Based Reasoning (CBR) systems to store the experience and provide this knowledge for diagnosis. In the next section, we give an overview of the OMAHA project and the SEASALT architecture. Section 2 contains related work with comparing our approach to other diagnosis and multi-agent approaches. In Section 3 we describe the instantiation of the SEASALT components for our MAS and describe the case-bases agents with the case structure and similarity measures of the underlying CBR systems in more detail. Section 4 gives a short summary of the paper and an outlook on future work.

In Proceedings of the ICCBR 2015 Workshops. Frankfurt, Germany.

1.1 OMAHA project

The multi-agent system for case-based aircraft diagnosis is under development in the context of the OMAHA research project. This project is supported by the German Federal Ministry of Economy and Energy and tries to develop an **O**verall **M**anagement **A**rchitecture for **H**ealth **A**nalysis of civilian aircraft. Several topics are addressed within the project like diagnosis and prognosis of flight control systems, innovative maintenance concepts, and effective methods of data processing and transmission. A special challenge of the OMAHA project is to integrate not only the aircraft and its subsystems, but also systems and processes in the ground segment like manufacturers, maintenance facilities, and service partners into the maintenance process. Several enterprises and academic and industrial research institutes take part in the OMAHA project: the aircraft manufacturer Airbus, the system manufacturers Diehl Aerospace and Nord-Micro, the aviation software solutions provider Linova and IT service provider Lufthansa Industrial Solutions as well as the German Research Center for Artificial Intelligence and the German Center for Aviation and Space. In addition, several universities are included as subcontractors. The project started in 2014 and will last until the end of March, 2017. [1]

The OMAHA project has several different sub-projects. Our work focuses on a sub-project to develop a so-called integrated system health monitoring (ISHM) for aircraft systems. The main goal is to improve the existing diagnostic approach to identify faults with root cause in more than a single subsystem (cross-system faults). Therefore, a multi-agent system (MAS) with several case-based agents will be implemented to integrate experience into the diagnostic process and provide more precise diagnoses for given faults.

1.2 SEASALT architecture

The SEASALT (**S**hared **E**xperience using an **A**gent-based **S**ystem **A**rchitecture **L**ayou**T**) architecture is a domain-independent architecture for extracting, analyzing, sharing, and providing experiences [4]. The architecture is based on the Collaborative Multi-Expert-System approach [1],[2] and combines several software engineering and artificial intelligence technologies to identify relevant information, process the experience and provide them via an interface. The knowledge modularization allows the compilation of comprehensive solutions and offers the ability of reusing partial case information in form of snippets.

The SEASALT architecture consists of five components: knowledge sources, knowledge formalization, knowledge provision, knowledge representation, and individualized knowledge. The knowledge sources component is responsible for extracting knowledge from external knowledge sources like databases or web pages and especially Web 2.0 platforms.

The knowledge formalization component is responsible for formalizing the extracted knowledge into a modular, structural representation. This formalization

---

[1] `www.dlr.de/lk/desktopdefault.aspx/tabid-4447/7274_read-39606`

is done by a knowledge engineer with the help of a so-called Apprentice Agent. This agent is trained by the knowledge engineer and can reduce the workload for the knowledge engineer.

The knowledge provision component contains the so-called Knowledge Line. The basic idea is a modularization of knowledge analogous to the modularization of software in product lines. The modularization is done among the individual topics that are represented within the knowledge domain. In this component a Coordination Agent is responsible for dividing a given query into several sub queries and pass them to the according Topic Agents. The agent combines the individual solutions to an overall solution, which is presented to the user. The Topic Agents can be any kind of information system or service. If a Topic Agent has a CBR system as knowledge source, the SEASALT architecture provides a Case Factory for the individual case maintenance [4],[3],[9].

The knowledge representation component contains the underlying knowledge models of the different agents and knowledge sources. The synchronization and matching of the individualized knowledge models improves the knowledge maintenance and the interoperability between the components. The individualized knowledge component contains the web-based user interfaces to enter a query and present the solution to the user.

1.3 Application domain: aircraft diagnosis

An aircraft is a highly complex machine consisting of a large number of subsystems that interact with each other, like hydraulic, cabin, ventilation, and landing gear. Each subsystem has a large number of components. The smallest component that can be replaced during maintenance is called Line Replacement Unit (LRU). The challenge is to find the root cause of a fault, because there could be more than one LRU causing the fault or a fault chain. In a fault chain, the first fault causes additional faults, which could also cause additional faults again. Faults are not limited to have their root cause in the subsystems that stated the fault, but the root cause can be found in a different subsystem. Therefore, a cross-system diagnosis is required to improve the precision of the diagnosis process.

In the next section we give an overview of some related work. In Section 3 we describe the multi-agent system concept and the instantiation of the SEASALT architecture. Section 3.3 describes the current status of our implementation. Finally a summary and outlook on future work is given.

2 Related Work

Decision support for diagnosis (and maintenance) in the aircraft domain means that a lot of engineering knowledge is available to support this process. In the past various diagnostic approaches tried to improve diagnosis and maintenance in this domain: among others case-based reasoning, rule-based reasoning, model-based reasoning, Bayesian belief networks, Fuzzy inference, neural networks, fault trees, trend analysis, and a lot of combinations. For OMAHA, that is OMAHA work package 230, the exploitation of available experiences as supplementation to other already used knowledge sources is of high priority. See also the work from Reuss et al.[10] for relating our approach with a selection of related other experience reusing diagnostic approaches: the British research project DAME [7] dealing with fault diagnosis and prognosis based on grid computing , Dynamic Case-Based Reasoning [13] learning also through statistic vectors containing abstract knowledge condensed from groups of similar cases, and the hybrid approach of Ferret and Glasgow [6] combining model-based and case-based reasoning.

For optimizing the relation between cost and benefit we decided to use the various available textual knowledge sources (cf. also Section 3). A recent overview of using textual sources for CBR is given in the textbook of Richter and Weber [12]. The paper of Reuss et al. [11] also gives an overview of some related approaches in this direction.

In addition to other specific characteristics of our approach one property differentiating it from many other (CBR) approaches is the fact that we develop a multi-agent system that applies a lot of CBR agents (among other) ones. The following approaches have in common that they also combine a multi-agent system approach with CBR. Researchers also dealing with CBR from different perspectives and trying to combine the specific insights to an improved overall approach are [16]. Of course, what makes our approach different here is that we are concerned with the development of concrete framework with existing applications. Corchado et al.[5] present in their work an architecture for integrating multi-agent systems, distributed services, and application for constructing Ambient Intelligence environments. Besides addressing a different domain and task this approach appears to be more open concerning the potential tasks agents can take over, while our approach is more focused in applying software engineering strategies for decomposing problems into sub-problems resulting in a distributed knowledge-based system. Zouhaire and his colleagues[17] developed a multi-agent system using dynamic case-based reasoning that learns from traces and is applied for (intelligent) tutoring. Our approach does not learn from traces but instead has to deal with a lot of technical knowledge and in addition has to solve critical problems. Srinivasan, Singh and Kumar[14] share with our approach that they develop a conceptual framework for decision support systems based on multi-agent systems and CBR systems. Our approach appears to be more on the side of integrating software engineering and artificial intelligence methods implementing concrete application systems, while the authors discuss how their framework influences decision support system in general. Khelassi[8] developed the IK-DCBRC system basing on a multi-agent architecture using a CBR approach with fuzzy-enhanced similarity assessment and being able to explain the results for different users. Our approach is not explanation-aware with respect to its current implementation status, however there is a conceptional extension of the SEASALT architecture (together with Thomas Roth-Berghofer and his research team) defined that includes explanation awareness. In addition, there are two PhD research projects ongoing focusing on explanation awareness. What also makes us different from Khelassis work is that our approach is embedded in an overall methodology resulting in a systematic process of how to develop an instance of our architecture with applications in travel medicine, technical diagnosis, and architectural design.

3 Multi-agent case-based diagnosis in the aircraft domain

In this section we describe the current version of our multi-agent system for case-based diagnosis. Based on the SEASALT architecture we describe the instantiation of the single components in context of our multi-agent system and the diagnosis workflow. In addition, we give an overview over the case structures and similarity measures of our case-based agents.

3.1 Multi-agent system for aircraft diagnosis

First we will describe the instantiation of our multi-agent system. The multi-agent system is an additional component of the diagnosis mechanism. It will not replace the existing rule-based diagnosis, but will extend the current diagnosis mechanism. The main component for our multi-agent aircraft diagnosis is the *knowledge provision* component. This component contains the Knowledge Line, which is responsible for providing a diagnosis for a given fault situation. The Knowledge Line consists of several topic agents with underlying CBR systems. The topic agents use the knowledge of their CBR systems to provide a part of the diagnosis. If only the knowledge of one topic agent is required, the topic agents delivers the complete diagnosis. There are several homogeneous teams of topic agents in the Knowledge Line, each responsible for diagnoses of an aircraft type (e.g., A320, A350, or A380). Each team has an additional agent, called solution agent to coordinate the topic agents and rank the individual solutions. Because each individual solution represents a possible diagnosis, a combination of solutions is not appropriate. The approach of separated agent teams for each aircraft type is based on the idea to split the knowledge into several smaller CBR systems. This way the number of cases for a retrieval and the maintenance effort for each system can be reduced. Nevertheless, for a diagnosis more than one agent team may be necessary. Therefore, a query can be distributed to several agent teams, either by default or if a diagnosis from the primary agent team for a query cannot provide a sufficient diagnosis. A coordination agent is responsible for coordinating the agent teams, distributing a query, and combining the team's solutions. The complete diagnosis process requires some more software agents that do not belong to the Knowledge Line itself: an interface agent, a composition agent, a knowledge map agent, and an output agent. The interface agent receives the query either from a web interface and/or a data warehouse. The main data source is a Post Flight Report (PFR) containing all the faults having occurred during the last flight of an aircraft. This PFR is based on the rule-based diagnosis in the aircraft. Each fault is represented as a so-called PFR item. Additional data like aircraft configuration, operational parameters (e.g., weather conditions, temperature, etc.), and logbook entries can be received, too. The PFR data and the additional data have to be correlated to assign the additional data to the corresponding PFR item. This task is done by the correlation agent. The extended PFR items are sent to the coordination agent. For each PFR item, a request to one or more agent teams is performed. To determine which topic agents of a team should be requested, a so-called Knowledge Map is used. This Knowledge Map contains information about existing agents and their dependencies and underlying CBR systems. The task to determine a so-called retrieval path (the topic agents to be requested and the sequence of retrievals) is done by a knowledge map agent. This agent has access to the general Knowledge Map and a CBR system, which stores past successful retrieval paths for given fault situations. The knowledge map agent uses the CBR system to retrieve the most similar retrieval paths and adapt the path to the new situation if necessary. Based on the determined retrieval path, the topic agents are requested and a ranked list of diagnoses is generated. The list of diagnoses is sent to the output agent. The output agent forwards the list to the web interface and the data warehouse. One more agent is located in the *knowledge provision* component. The so-called query analyzer takes each extended PFR item and checks for new concepts, which are not yet part of the vocabulary of the CBR systems. If any new concepts are found, a maintenance request is sent to the so-called Case Factory [9]. The Case Factory checks the maintenance request, derives appropriate maintenance actions, and executes the actions after confirmation from a knowledge engineer. The query analyzer is not part of the diagnosis process itself, but provides some learning capabilities to the multi-agent system.

The user interface can be found in the *individualized knowledge* component. The user interface is a web interface, which provides the options to send a query to the multi-agent system and present the returned diagnoses. In addition, the user can enter new cases, edit existing cases, and browse a entire selected case base. In addition to the web interface, a connection to a data warehouse is part of this component. The data warehouse contains PFRs and the additional data and will be the main query source for the multi-agent systems. If additional information is required that is not provided by the data warehouse, it can be added via the web interface.

The *knowledge formalization* component transforms structured, semi structured, and unstructured data into a modular, structural knowledge representation used by all CBR systems. This way the knowledge is represented in the same way all over the multi-agent system. Because a structural approach for the CBR systems in the Knowledge Line was chosen, semi-structured and unstructured data have to be transformed into attribute value pairs. This transformation workflow is performed by a so-called case base input analyzer. The workflow consists of several steps: At first, information extraction methods are used to extract keywords and collocations and to find synonyms and hypernyms for the extracted keywords. Then the input data is analyzed to find associations within the allowed values of an attribute as well as across different attributes.

This way want to generate completion rules for query expansion. The keywords, synonyms, hypernyms, and collocations are added to the vocabulary and initial similarity values for keywords and their synonyms are set. The keywords and their hypernyms can be used to generate taxonomies for similarity measures. After the vocabulary extension, cases are generated from the input data and stored in the case bases. The last step is to perform a sensitivity analysis on the stored cases to determine the weighting for the problem description attributes. The workflow is presented in more detail in [11].

In the *knowledge sources* component a collector agent is responsible for finding new data in the data warehouse, via web services or in existing knowledge sources of Airbus. New data could be new configurations or operational parameters, new synonyms or hypernyms, or complete new cases.

The *knowledge representation* component contains the generated vocabulary, similarity measures and taxonomies, completion rules, and constraints provided for all agents and CBR systems.

3.2 Case-based agents

This section focuses on the case-based agents within our multi-agent diagnosis system. We will describe the agents' tasks and the underlying CBR systems with their case structure and similarity modeling. In addition to the PFR, we have to consider several different data structures like Service Information Letters (SIL), In-Service Reports (ISR), elogbooks and aircraft configuration documents. While a PFR contains only information about the problem description, SIL, ISR and eLogbooks contain problem descriptions and solutions. Configuration documents contain data about the latest system configuration of an aircraft with hard- and software versions. We performed an analysis on these data to identify relevant information for cases, relationships between these information and data anomalies. Based on the result of this analysis we derived two case structures with attribute-value pairs and their value ranges. One case structure is based on PFR and SIL (C_{SIL}) and the other case structure is based on PFR and ISR (C_{ISR}). The case structures overlap to some degree, because attributes derived from the PFRs are part of both structures, like ATA chapter, aircraft type, and fault description. The C_{SIL} structure contains 32 attributes, while the C_{ISR} structure consists of 28 attributes. The attributes are distributed among problem description, diagnosis, quality information, and pointer to other cases. The problem description contains attributes like ATA chapter, aircraft type (e.g., A380), aircraft model (e.g., 380-641), fault code, displayed message, fault description and affected Line Replacement Units (LRU). Attributes like recommendation, comments, maintenance reference, corrective LRUs and root cause are part of the solution. For quality assessments the number of positive (a retrieved diagnosis was helpful) and negative (a retrieved diagnosis was not helpful) retrievals are stored.

The configuration of an aircraft has great impact on the probability of fault occurrence. If a certain system is not built in, corresponding faults will not occur. The occurrence of faults depends also on the soft- and hardware version of built in systems. Therefore, the configuration of an aircraft can exclude faults and root causes and have an impact on the similarity of cases. Because of the complexity of the configuration data for an entire aircraft, we decided to consider the configuration separate for each aircraft component. For each subsystem of a component the so-called modification status (mod-status) is stored. With the help of this mod-stati, cases could be excluded and similar configuration could be compared.

Most of the attributes have a symbolic data type and a taxonomy as similarity measure. The attributes ATA chapter, fault code and affected LRUs have a natural hierarchical structure, that can be mapped to a taxonomy. A great challenge is the similarity measure for the fault description attribute. The symbolic values of this attribute are extracted via a workflow in the knowledge formalization component as described in [11]. During the automatic vocabulary expansion, the values are added to a similarity table. Similarities between the automatically added values are only set between values and their synonyms. The other values have to be set manually. To reduce the effort, an automatic taxonomy generation from the extracted values and their synonyms and hypernyms is planned.

The multi-agent system will contain several topic agents with the same case structure to reduce the number of cases in one case base. Most faults can be assigned to a specific ATA chapter. Therefore, for each ATA chapter an own topic agent is generated. An agent team within our multi-agent system will consist of agents discriminated by ATA chapter and data source (SIL, ISR, etc.).

Another case-based agent is the so-called knowledge map agent. This agent is responsible for determining which topic agents have to be requested to find a solution for a given request. For each request, a retrieval on the underlying CBR system is performed. The cases will contain the characteristics of a request as the problem description and a successfully used retrieval path. This way we try to address the cross-system faults. Cross-system faults may have their root causes in LRUs of different ATA chapters. Requesting only the topic agent of a single ATA chapter may not give the correct root cause identification and diagnosis. Based on experience from solved faults, the cases for the knowledge map agent could contain information when the request of additional topic agents may be useful to find the correct diagnosis.

There are several challenges to be met while modeling the case structures and the similarity measures. One major challenge is based on the fact, that the ATA chapter differs for the same subsystem in different aircraft types. The cabin entertainment system is linked to two different ATA chapters in the A320 and the A380. Therefore, a mapping between the different ATA chapters is required to compare fault cases from different aircraft types. Another challenge is modeling the fault description in the case structure. The description of a fault is mainly given in free text provided by pilots or cabin personal. Unfortunately, there is no standard description language for faults. Therefore, every person describes a fault with slightly different words and technical terms. Extracting the key symptoms from this fault descriptions and comparing two fault descriptions requires the integration of natural language processing techniques in the modeling process and the diagnosis process of the multi-agent system. In addition, the amount of knowledge that can be found in the fault descriptions is very high. Analyzing 3000 example fault descriptions, we found more than 21000 different keywords and phrases describing the occurred faults. Modeling all these keywords and phrases in one attribute is not practicable. While it is possible to add all keywords automatically, setting the similarity between these keywords within a similarity matrix or a taxonomy is not practicable. In addition, the maintenance effort for such an attribute would be very high and in no relation the gained benefit.

The main challenge for the knowledge map agent is to identify the characteristics of a request and the according knowledge sources to solve the request.

3.3 Status of implementation

We implemented a prototype to test some functionalities of the desired multi-agent system. This application serves as a testing system for knowledge modeling and diagnosis process. The prototype consists of two CBR systems and a user interface to interact with the systems. We modeled the case structure, vocabulary and similarity using the open source tool myCBR [15]. One CBR system contains cases based on SIL documents, the other one on ISR documents. The SIL case base contains 670 cases and the ISR case base 220 cases. The user interface provides the functionalities to perform a retrieval, enter new cases, edit existing cases, and browse the case base based on filter criteria. In addition, the workflow of the knowledge extraction is partially implemented. The keyword extraction, collocation extraction, synonym and hypernym identification, and automatic vocabulary extension are implemented. For more detail on the implementation of the knowledge extraction workflow see [11].

4 Summary and Outlook

In this paper we describe the instantiation of our multi-agent system for case-based diagnosis. We give an overview of the individual components and describe the case structure and similarity of our case-based agents. As Section 3.3 shows, the multi-agent system is not fully implemented, yet. The next steps are the implementation of the additional agents (interface, coordination, output, knowledge map) and the refinement of the case structures and similarity measures. In addition, the learning mechanism based on the knowledge extraction workflow will be realized.

References

1. Althoff, K.D.: Collaborative multi-expert-systems. In: Proceedings of the 16th UK Workshop on Case-Based Reasoning (UKCBR-2012), located at SGAI International Conference on Artificial Intelligence, December 13, Cambride, United Kingdom. pp. 1–1 (2012)

2. Althoff, K.D., Bach, K., Deutsch, J.O., Hanft, A., Mänz, J., Müller, T., Newo, R., Reichle, M., Schaaf, M., Weis, K.H.: Collaborative multi-expert-systems – realizing knowledge-product-lines with case factories and distributed learning systems. In: Baumeister, J., Seipel, D. (eds.) KESE @ KI 2007. Osnabrück (Sep 2007)
3. Althoff, K.D., Hanft, A., Schaaf, M.: Case factory - maintaining experience to learn. Advances in Case-Based Reasoning Lecture Notes in Computer Science 4106/2006, 429–442 (2006)
4. Bach, K.: Knowledge Acquisition for Case-Based Reasoning Systems. Ph.D. thesis, University of Hildesheim (2013), dr. Hut Verlag Mnchen
5. Corchado, J.M., Tapia, D.I., Bajo, J.: A multi-agent architecture for distributed services and applications. International Journal of Innovate Computing 8, 2453–2476 (2012)
6. Feret, M., Glasgow, J.: Combining case-based and model-based reasoning for the diagnosis of complex devices. Applied Intelligence 7, 57–78 (1997)
7. Jackson, T., Austin, J., Fletcher, M., Jessop, M.: Delivering a grid enabled distributed aircraft maintenance environment (dame). Tech. rep., University of York (2003)
8. Khelassi, A.: Reasoning System for Computer Aided Daignosis with explanation aware computing for medical applications. Ph.D. thesis, Abou Bakre Belkaied University, Tlemcen, Algeria (2013)
9. Reuss, P., Althoff, K.D.: Explanation-aware maintenance of distributed case-based reasoning systems. In: LWA 2013. Learning, Knowledge, Adaptation. Workshop Proceedings. pp. 231–325 (2013)
10. Reuss, P., Althoff, K.D., Henkel, W., Pfeiffer, M.: Case-based agents within the omaha project. In: Case-based Agents. ICCBR Workshop on Case-based Agents (ICCBR-CBR-14) (2014)
11. Reuss, P., Althoff, K.D., Henkel, W., Pfeiffer, M., Hankel, O., Pick, R.: Semi-automatic knowledge extraction from semi-structured and unstructured data within the omaha project. In: Proceedings of the 23rd International Conference on Case-Based Reasoning (2015)
12. Richter, M.M., Weber, R.: Case-Based Reasoning - A Textbook. Springer-Verlag Berlin Heidelberg (2002)
13. Saxena, A., Wu, B., Vachtsevanos, G.: Integrated diagnosis and prognosis architecture for fleet vehicles using dynamic case-based reasoning. In: Autotestcon 2005 (2005)
14. Srinivasan, S., Singh, J., Kumar, V.: Multi-agent based decision support system using data mining and case based reasoning. International Journal of Computer Science Issues 8, 340–349 (2011)
15. Stahl, A., Roth-Berghofer, T.: Rapid prototyping of cbr applications with the open source tool mycbr. In: Advance in Case-Based Reasoning, Proceeding of the 9th European Conference on Case-Based Reasoning (2008)
16. Sun, Z., Han, J., Dong, D.: Five perspectives on case based reasoning. In: 4th International Conference on Intelligent Computing. pp. 410–419 (2008)
17. Zouhair, A., En-Naimi, E.M., Amami, B., Boukachour, H., Person, P., Bertelle, C.: Incremental dynamic case based reasoning and multi-agent systems (idcbr-mas) for intelligent touring system. International Journal of Advanced Research in Computer Science and Software Engineering 3, 48–56 (2013)

A Case-Based Framework for Task Demonstration Storage and Adaptation

Tesca Fitzgerald, Ashok Goel

School of Interactive Computing,
Georgia Institute of Technology,
Atlanta, Georgia, 30332
`{tesca.fitzgerald,goel}@cc.gatech.edu`

Abstract. We address the problem of imitation learning in interactive robots which learn from task demonstrations. Many current approaches to interactive robot learning are performed over a set of demonstrations, where the robot observes several demonstrations of the same task and then creates a generalized model. In contrast, we aim to enable a robot to learn from individual demonstrations, each of which are stored in the robot's memory as source cases. When the robot is later tasked with repeating a task in a new environment containing a different set of objects, features, or a new object configuration, the robot would then use a case-based reasoning framework to retrieve, adapt, and execute the source case demonstration in the new environment. We describe our ongoing work to implement this case-based framework for imitation learning in robotic agents.

Keywords: Case-based agents, imitation learning, robotics

1 Introduction

Imitation is an essential process in human social learning and cognition [11, 10]. Imitation learning occurs when a learner observes a teacher demonstrating some action, providing knowledge of (i) how the action was performed and (ii) the resulting effects of that action. This interaction-guided learning method allows us to learn quickly and effectively. As a result of its importance in human cognition, it follows that imitation learning has become an area of increasing focus in interactive robotics research as well.

The goal of *Learning from Demonstration* is to enable imitation learning in robots through interactive demonstrations, provided through methods such as teleoperation or kinesthetic teaching [1, 2]. Regardless of which demonstration method is used, the following process is often used. First, the human teacher provides several demonstrations of a skill. Between demonstrations, the teacher may adjust the environment such that the skill is demonstrated in a variety of initial configurations. The robot then creates an action model which generalizes over the provided demonstrations. Lastly, the robot applies the generalized action model to plan a trajectory which is executed in a new environment.

In Proceedings of the ICCBR 2015 Workshops. Frankfurt, Germany.

However, a challenge of this process is that the resulting action model is dependent on the number of demonstrations that were provided for that particular task. We also assume that the robot has been exposed to enough variations of the initial configuration such that its generalized model can be applied to a wide range of related initial configurations. As such, the generalized model is restricted to application in environments which are similar to those demonstrated.

We describe our preliminary work toward defining an alternate approach to imitation learning in robotics, one which takes a *case-based* approach in which the robot stores demonstrations *individually* in memory. We define a case-based framework which enables the full imitation learning process, from observing a task demonstration to transfer and execution. We also define a case representation which encodes task demonstrations for storage in source case memory.

2 Related Work

Case-based reasoning has been used to address the problem of transfer in robotics domains. Floyd, Esfandiari & Lam [7] describe a CBR approach to learning strategies for RoboCup soccer by observing spatially distributed soccer team plays. Their approach represents each case as an encoding of a single agent's perception and resulting action at a given time. Thus, they transfer the behavior of an agent when it perceives a situation similar to that of the observed agent. More recently, Floyd & Esfandiari [6] describe an approach for case-based learning by observation in which strategy-level domain-independent knowledge is separated from low-level, domain-dependent information such as the sensors and effectors on a physical robot. Ontañón et al. [8] describe their approach to observational learning for agents in real-time strategy games. They use a case-based approach to online planning, in which agents adapt action plans which are observed from game logs of expert demonstrations.

While these approaches do address knowledge transfer for robotic and simulated agents, they primarily operate over input and output represented at a higher level of abstraction, such as actions at a strategic level. The goal of our work is to enable transfer to generate action at a lower level of control and in response to real-world perceptual input, where we transfer the demonstrated action trajectory used to achieve a task. We expand on our previous work [3] describing a case-based approach to interpretation and imitation in robotic agents. We discussed two separate processes: (i) interpreting new skill demonstrations by comparing it to previously observed demonstrations using a case based process (further described in [5]), and (ii) a related process for imitating a task demonstration. This paper expands on the latter process, case-based imitation.

We previously provided a general outline for imitation in [3] in which four steps occur: representation of the task demonstration at multiple levels of abstraction, retrieval of the most relevant source case from memory, adaptation of the source case to address the target problem, and execution of the adapted case in the target problem. In this paper, we describe our more recent work providing (i) a revised, complete process of imitation beginning with observation

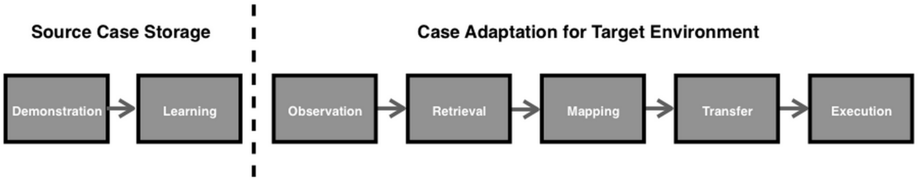


Fig. 1. Case-Based Process for Task Demonstration Transfer

of the task demonstration and ending with task transfer and execution, (ii) a *Mapping* step which bridges the gap between the *Retrieval* and *Transfer* steps, and (iii) a revised case representation for storing task demonstrations (iterating on preliminary work introduced in [4]).

3 Approach Overview

We have revised our case-based approach to transfer (originally summarized in [3]) to consist of two separate processes, as shown in Figure 1: the *Case Storage* process in which the robot receives demonstrations of a task and stores each demonstration as a case in source memory, and a *Case Adaptation* process which is used at a later time when the robot is asked to repeat a task in a target environment.

3.1 Why a CBR approach?

Our eventual goal is to enable transfer for imitation learning in scenarios such as the following. A human teacher guides the robot to complete a task such as scooping the contents of one container into another. During the demonstration, the robot records the demonstrated trajectories and object features. At a later time, the robot is asked to repeat the *scooping* task, but in a new, *target* environment. Thus, the robot must use a different set of object features to parameterize and execute the *scooping* task than those observed in the original, *source* environment. Next, the robot transfers its representation of the *scooping* task to accommodate for the differences between the source and target environments. The transferred task representation is then executed in the target environment.

Rather than generalize over a set of demonstrations as in current Learning from Demonstration methods (surveyed in [1, 2]), using a case-based approach allows us to: (1) operate under the assumption that the human teacher will provide a limited number of demonstrations, (2) represent demonstrations as individual experiences in the robot's memory, and (3) utilize a complete framework for transferring skill demonstrations, which includes the steps of retrieving, analyzing, transferring, and executing a relevant *source case* demonstration in an unfamiliar, *target environment*.

3.2 Case Storage Process

Demonstration and Learning We have implemented the first step in the *Case Storage* process, where the robot records and stores each task demonstration as a source case in memory. We define each case as the tuple $C = <L, D, T, O, S_i, S_f>$, where:

- L represents the label of the task which was demonstrated, e.g. "scooping".
- D represents the set of action models which encode the demonstrated motion, represented as Dynamic Movement Primitives as defined in [9].
- T is the set of parameterization functions which relate the set of action models to the locations of objects in the robot's environment. For example, a parameterization function may be used to represent how the robot's hand must be located above a bowl prior to completing a *pouring* action.
- O is the set of *salient* object IDs which are relevant to the task.
- S_i and S_f are the initial and final states, respectively, which represent the set of objects observed in an overhead view of the robot's environment.

3.3 Case Adaptation Process

At a later time, the robot may be asked to repeat the task in a new, target environment. We are currently implementing the Case Adaptation process shown in Figure 1.

Observation will begin when the robot is asked to address a target problem. We assume that the robot has been provided a relevant source case which it can retrieve from memory to address the given target problem. The robot will then observe the target environment by viewing the objects located in the table-top environment using an overhead camera. This will provide it with the target case's initial state S_i.

Retrieval must be performed to select a source case from memory containing the demonstration that is most relevant to the current target problem. Case retrieval will prioritize (i) similarity of task goals, (ii) similarity of salient objects, and finally, (iii) similarity of initial states. Once a relevant source case has been retrieved, the *Mapping* step must encode the differences between the source and target environments. This mapping will be later used to transfer the source case such that differences in the target environment are addressed.

Given a source case and mapping which encodes the differences between the source and target cases, the *Transfer* step adapts the source case. We take a similarity-based approach to transfer, where we consider the similarity between the source case and target environments when defining transfer processes. As we encounter transfer problems in which the source and target problems become less similar, the source case is transferred at a different level of abstraction, such that only high-level features of that case are transferred. The adapted case is then *executed* in the target environment.

We have implemented three methods which implement the *Transfer* step, each of which operates by transferring the source case at a different level of abstraction. Once the source case has been transferred, it is used to plan and execute a new action trajectory. In preliminary experiments, we have evaluated each method separately such that we selected the level of abstraction at which transfer occurred in each target problem. These experiments have shown us that by changing the level of abstraction at which a case is transferred, a robot can use a single source demonstration to address target environments of varying similarity to the source environment.

4 Future Work

We have implemented the Case Storage process and the last two steps of the Case Adaptation process, the *Transfer* and *Execution* steps. Currently, we manually provide the robot with the most relevant source case demonstration and a mapping between objects in the source and target environments. Thus, our next steps are to identify a method for autonomously determining this object mapping. Furthermore, future work will involve defining a process for identifying and retrieving an appropriate source case demonstration that is most applicable to a given transfer problem.

Acknowledgments

This work was supported by NSF Graduate Research Fellowship DGE-1148903.

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Autonomous Systems 57(5), 469–483 (2009)
2. Chernova, S., Thomaz, A.L.: Robot learning from human teachers. Synthesis Lectures on Artificial Intelligence and Machine Learning 8(3), 1–121 (2014)
3. Fitzgerald, T., Goel, A.: A case-based approach to imitation learning in robotic agents. Intl. Conf. on Case-Based Reasoning Workshop on Case-Based Agents (2014)
4. Fitzgerald, T., Goel, A.K., Thomaz, A.L.: Representing skill demonstrations for adaptation and transfer. AAAI Symposium on Knowledge, Skill, and Behavior Transfer in Autonomous Robots (2014)
5. Fitzgerald, T., McGreggor, K., Akgun, B., Thomaz, A.L., Goel, A.K.: Visual case retrieval for interpreting skill demonstrations. International Conference on Case-Based Reasoning (2015)
6. Floyd, M.W., Esfandiari, B.: A case-based reasoning framework for developing agents using learning by observation. In: 2011 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI). pp. 531–538. IEEE (2011)
7. Floyd, M.W., Esfandiari, B., Lam, K.: A case-based reasoning approach to imitating robocup players. In: FLAIRS Conference. pp. 251–256 (2008)
8. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-based planning and execution for real-time strategy games. In: Case-Based Reasoning Research and Development, pp. 164–178. Springer (2007)
9. Pastor, P., Hoffmann, H., Asfour, T., Schaal, S.: Learning and generalization of motor skills by learning from demonstration. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. pp. 763–768. IEEE (2009)
10. Piaget, J., Cook, M.T.: The origins of intelligence in children. (1952)
11. Tomasello, M., Kruger, A.C., Ratner, H.H.: Cultural learning. Behavioral and brain sciences 16(03), 495–511 (1993)

Case Based Disruption Monitoring

Joe Kann, Matthew Molineaux and Bryan Auslander

Knexus Research Corp.
174 Waterfront Street, Suite 310, National Harbor, MD 20745

`firstname.lastname@knexusresearch.com`

Abstract. Mine Countermeasures Missions (MCM) take place in very complex and uncertain environments which poses complexity for planning and explanation algorithms. In order to keep a mission on target, constant disruption monitoring and frequent schedule adjustments are needed. To address this capability gap, we have developed the Case-Based Disruption Monitoring and Analyzing (CDMA) algorithm. The CDMA algorithm automatically detects disruptions within a mission and attempts to determine possible root causes. Once confirmed, our second developed algorithm, CLOSR modifies existing schedules to compensate for these root causes. Evaluation of CDMA on simulated MCM operations demonstrates the effectiveness of case-based disruption monitoring. Both the CDMA and CLOSR algorithms, along with simulator, are enclosed with our KRePE system.

1 Introduction

Unforeseen disruptions occur when planning in the real world. When monitoring for such disruptions and providing an explanation as to why the disruption occurs, better insight is provided in order to fix the plan. Mine Countermeasure Missions (MCM) for example, uses planning constantly. MCM planning uses a variety of resources and each resource has its own set of capabilities and operational constraints, as well as characteristic failure points.

Mine Countermeasure Missions (MCM) must respond to frequent disruptions, and recovering from these disruptions can be complex. MCM missions involve the location, identification, and neutralization of enemy explosive ordnance in a maritime context. This is key to naval power projection and sea control, two core capabilities of U.S. maritime power, as characterized by *A Cooperative Strategy for 21st Century Seapower* [4]. Due to high complexity and uncertainty when scheduling MCM missions, accurate plans must be created and frequently revised once a mission has started. Frequent disruptions in MCM operations can occur due to many types such as: changes in sea state, visibility, weather, equipment failure, etc. Situations like these interfere with resource availability and/or readiness. Therefore, schedules for MCM operations require frequent changes and updates where the disruptions are monitored in order to keep the success of the mission. Current practice calls for manually observing all incoming data

In Proceedings of the ICCBR 2015 Workshops. Frankfurt, Germany.

for detection of issues that could cause a mission to fail. The manual process of monitoring for disruptions can be tedious and prone to error.

To meet this need, we are developing a system for MCM operation decision making and planning support called KRePE. KRePE builds upon a foundation of cognitive architecture components, algorithms and simulations. Housed within the KRePE architecture the Case-Based Disruption Monitoring and Analyzing (CDMA) algorithm performs monitoring and analysis of disruptions and Case-Based Local Schedule Repair (CLOSR) reschedules tasks that MCM planner operators perform on a frequent basis. Both the CDMA and CLOSR algorithms fall in a problem solving paradigm known as Case-based reasoning (CBR) by relying on general and specific knowledge of MCM operations, how operations might be disrupted, and how to fix these interruptions.

In this paper, we discuss the challenges of continuous situation monitoring, and root cause analysis of mission disruptions through case-based reasoning. We close with an empirical study that demonstrates this effective anomaly detection in order to generate schedule modifications that achieve mission success.

2 Mine Countermeasures Mission Scheduling & Operations

MCM operations involve the location, identification, and neutralization of sea mines [5]. These operations employ surface vehicles, aircraft, divers, and unmanned surface and underwater vehicles, and can take weeks to plan and execute. While the operations are taking place, they are disrupted early and often by events such as unforeseen weather conditions, technological failures, and incorrect enemy course of action estimations. While technology exists to automatically create an initial schedule, distribute tasks, and track task completion, the critical monitoring and rescheduling tasks have been, to date, poorly supported [6].

MCM operations involve a unique set of specialized tasks that must be scheduled to minimize the risk to ships from sea mines. What follows is a brief description of the tasks in an MCM operation and their characteristics. The schedule for an MCM operation tasks multiple vehicles to repeatedly *hunt* and/or *sweep* subsections of a specified *threat area* where mines are expected, slowly transiting back and forth in a lawnmower-like search pattern, until the risk of remaining mines is reduced to an acceptably low level. The paths followed by these search vehicles are referred to as *tracks*.

Hunting is a search and destroy activity that encompasses use of specialized sensors to find underwater objects that are *mine-like*, identification of mine-like objects as *mines* or *non-mines*, and neutralization of all discovered mines. The *probability of detection* describes the equipment's sensitivity within that range to the size and reflectivity of mine casings. Because mines may be missed, missions are commonly evaluated according to a *percent clearance* objective. Percent clearance is defined as the probability that a mine at any given position in the search area will be detected.

Sweeping is an activity that uses specialized apparatus to destroy all mines present in a given area either by cutting the chains that connect them to the ocean floor or employing signal generators which mimic the magnetic and acoustic signatures, of ships, to trigger mines that are activated by those signatures.

The operation schedule, which may consist of hundreds of tasks of heterogeneous types, must be repeatedly adjusted over the course of the operation in response to unexpected events which invalidate it. The task of keeping the schedule up to date despite hundreds of interrelated tasks is complex, difficult, and laborious, particularly given the constant time pressure of typical operations. Modifications to schedules are kept to a minimum, in order to reduce expense and opportunities for error; we refer to this characteristic as *minimal operational disruption*. However, modified schedules must also fulfill operational requirements such as percent clearance, time limits, and risk to equipment. These difficult tasks (i.e., monitoring, response, and rescheduling) can be greatly aided by new computational tools.

3 CDMA

One way to reduce the burden on MCM human operators is to help with constant monitoring of disruptions that will impact the mission. Constant monitoring of a vast array of disruption types can be quite difficult. In addition to detecting the disruption, diagnosing the root cause of the problem can be daunting, or easily overlooked. Case-Based Disruption Monitoring and Analyzing (CDMA) within the KRePE architecture handles both disruption monitoring and providing possible root causes.

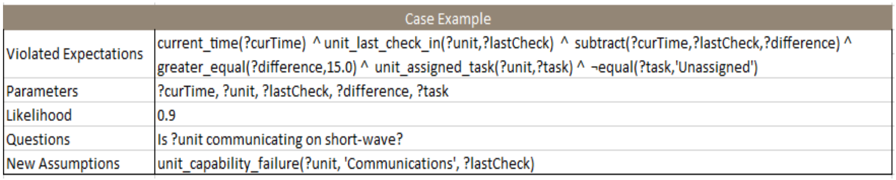
Case-based reasoning (CBR) is a problem solving paradigm that relies on general cases of a problem domain along with specific domain cases. These cases consist of a mapping between problems and a solution. When a new problem is introduced, generally CBR systems map and provides this new problem to the most similar problem already stored in its case base and provides a solution associated with the known problem. We describe the case representation and the CDMA algorithm in detail in the following subsections.

3.1 CDMA Representation

CDMA uses case-based reasoning for monitoring and analysis of disruptions that will impact an ongoing operation. Based on limited information of the world state, the CDMA algorithm determines if a disruption has occurred. A disruption case in our system are generated manually and consists of five parts: violated expectations, parameters, root cause likelihood, root cause questions and new assumptions.

The case applies when all of the violated expectations are met; and the parameters indicate which variables are applied to a specific problem instance. An example problem representation is shown in Table 1. In this example, there is a disruption where the operator has not heard from the unit within the past 15 minutes while it was out in the field performing a task.

The likelihood and list of root cause questions provide information that can be accessed by an operator through an interactive decision making process. The likelihood provides an apriori probability of how likely a particular root cause is for a given disruption. The root cause tests constitute a set of questions that can help the operator deduce what is causing the disruption. The parameters defined by the violated expectations populate the variables within the questions, detailing the questions to a specific unit, piece of equipment, etc. If these questions are answered, the likelihoods for the root causes adjust to this information. Using the example from above, Table 1 provides the entire case representation. The new assumptions are a set of suppositions or beliefs as to which root cause explains the disruption. The parameters defined from the violated expectations instantiate the problem information into these new assumptions.



	Case Example
Violated Expectations	current_time(?curTime) ^ unit_last_check_in(?unit,?lastCheck) ^ subtract(?curTime,?lastCheck,?difference) ^ greater_equal(?difference,15.0) ^ unit_assigned_task(?unit,?task) ^ ¬equal(?task,'Unassigned')
Parameters	?curTime, ?unit, ?lastCheck, ?difference, ?task
Likelihood	0.9
Questions	Is ?unit communicating on short-wave?
New Assumptions	unit_capability_failure(?unit, 'Communications', ?lastCheck)

Table 1. Case Representation for CDMA algorithm.

With the use of a standard relational database called the Integrated Rule Inference System (IRIS) [8], CDMA can reuse case(s) in the problem space without having to generate new cases for each set of parameter values. Therefore similarity metrics are not being used. From the example, we do not need to create new cases for each type of equipment or unit, as it can handle all of the parameters. When monitoring detects a disruption, it alerts human operators with a message. The operator then decides the root cause of a given disruption. CDMA adds this confirmed root cause assumptions to the case base providing more information to its case base. These new assumptions trigger schedule repair to occur because the disruption affects the mission.

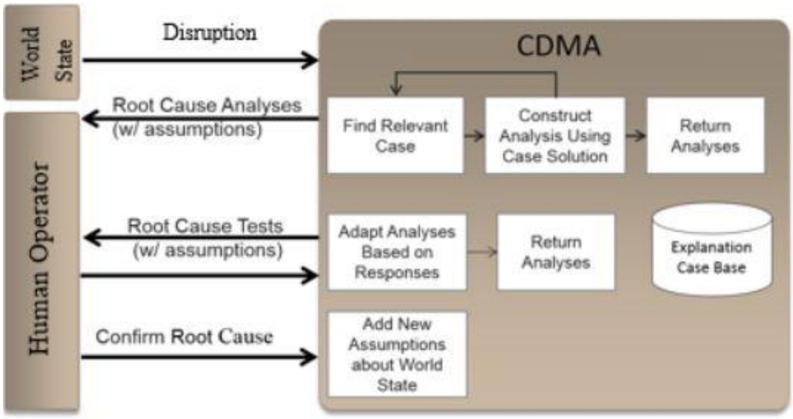

World State
Disruption
CDMA
Root Cause Analyses (w/ assumptions)
Find Relevant Case
Construct Analysis Using Case Solution
Return Analyses
Human Operator
Root Cause Tests (w/ assumptions)
Adapt Analyses Based on Responses
Return Analyses
Explanation Case Base
Confirm Root Cause
Add New Assumptions about World State

Fig. 1. Workflow for CDMA algorithm.

3.2 CDMA Algorithm

CDMA performs the following steps for disruption monitoring and analysis as shown in Figure 1:

1. ***Find Relevant Case***: To find a possible disruption, CDMA searches through the list of cases to find a relevant case that matches a violated expectation. Each case that matches provides a possible root cause for the disruption.
2. ***Construct Analysis Using Case Solution***: To analyze a disruption, the parameter values indicated by a specific violated expectation are substituted for the parameters specified by an individual case problem.
3. ***Return Analyses***: Each possible disruption is provided on screen for the user to review, detailing the types of root causes for the disruption, along with additional information such as root cause tests and likelihood for each cause.
4. ***Adapt Analyses Based on Responses***: Users can answer these root cause test questions in order for the system to better understand the disruption for future root causes.
5. ***Return Analyses***: The system returns updated likelihoods, sorted with highest likelihood first, along with clearing out infeasible causes.
6. ***Add New Assumptions about World State***: After user selection of the root cause for a disruption, the system creates new assumptions about the world and why the disruption occurred. These new assumptions are added into the case base, providing new information that can be used to generate schedule repair if necessary.

4 CLOSR

To repair schedules that don’t meet the criterion of minimal operation disruption, we use the Case-Based Local Schedule Repair (CLOSR) algorithm [10]. This Case base reasoning algorithm in the KRePE architecture creates new assumptions and generates repairs. These repairs strive for “minimal disruption” meaning changes to the schedule should be kept at a minimum while rescheduling to fix a disruption. For example, in MCM operations, repairing a vehicle communication disruption might try to resolve the problem without leaving its search area to minimize transiting time and fuel. Subsequent to case reuse, an adaptation process examines and resolves conflicts created by the schedule repair procedure which is useful for its flexibility. For more detail, please see [10].

5 Evaluation

We hypothesize that the discrepancy monitoring and analysis capabilities of CDMA outperforms ablations that ignore alerts or acts on randomly-selected root causes. To demonstrate this, we ran the CDMA algorithm in an automated manner on a series of simulated MCM operations. For each operation, we measured and compared the performances of three decision makers that: (1) ignores all alerts from CDMA and keeps the original schedule, (2) acknowledges CDMA found disruptions and chooses a random root cause from those suggested therefore rescheduling randomly and (3) acknowledges CDMA found disruptions and chooses the root cause with the highest likelihood. Difference between decision makers indicate the performance improvement that can be achieved by adopting the recommendations made by the CDMA algorithm.

Our study examines an MCM mission with a mine clearing objective. As it is impossible to ensure that 100% of mines are removed in the real world, missions are planned to achieve a high level of percent clearance. This means that there is a high chance that a mine at any given point in the search area would be observed if it existed. The operations conducted in our study are intended to achieve a 95% clearance level; in other words, we would expect 95% of the mines present to be removed. We hypothesize that the decision maker using KRePE's case base will achieve these performance objectives, and that the decision maker that ignores the disruptions will not. This will demonstrate both that monitoring and analyzing disruptions is necessary to achieve an acceptable level of performance under simulated conditions, and that the system is sufficient to achieve that performance.

5.1 Experimental Framework

A simulator for MCM operations, Search and Coverage Simulator (SCSim), another component of KRePE, supports rapid and repeated evaluation and testing of MCM decision support systems and component algorithms. SCSim simulates search missions involving multiple heterogeneous search units, including ships and helicopters, each with different available equipment configurations. Mines and mine-like objects are distributed randomly by SCSim in fields and lines according to pre-set distributions with variable density and object counts. This facilitates evaluation of algorithm performance under varying operating conditions. As a benchmark, automated testing of a two month operation takes less than one minute.

SCSim simulates the assignment of parameterized tasks to units according to a schedule, including transit, sweep, and hunt tasks. Task parameters include, for example, the equipment to use for sweeping, and sensor depth for hunting. To simulate a mission, SCSim automatically generates appropriate tracks for each task and simultaneously changes the position of each vehicle along its assigned tracks. Observations (e.g., contacts) are generated based on vehicles' positions and the sensor equipment in use. Interactions of deployed sweeping equipment is also simulated, and changes the internally represented status of mines. In addition to the scheduled tasks, SCSim is responsible for simulating random events the unexpected difficulties that invalidate an existing schedule (e.g., equipment failure, bad weather, operator errors).

An individual mission test using SCSim is controlled by a scenario description. Scenario descriptions include, at a minimum, the vehicles and equipment available for use, threat areas to be cleared of sea mines, and task areas where vehicles will operate. Other elements of the scenario specify random distributions for mine like objects, mine line placements, and events that may occur. To mimic the real world as closely as possible, SCSim provides only partial observations for the purposes of rescheduling. For example, when a helicopter's communications system fails, its position is no longer reported to the system. As a result, the helicopter appears not to move.

Experiments are driven by a test harness that integrates with SCSim as shown in Figure 2. The test harness generates scenarios defining: the area of operations, available assets, and the ranges of random experimental variables, such as what mine types will be deployed and when events will trigger. The Test Generator applies an appropriate decision maker that acts as a user of the system. Each decision maker encodes different responses to situations, such as alerts, that arise during the mission simulation. After all simulated missions are complete, the Performance Evaluator tabulates and summarizes these results in a human readable form.

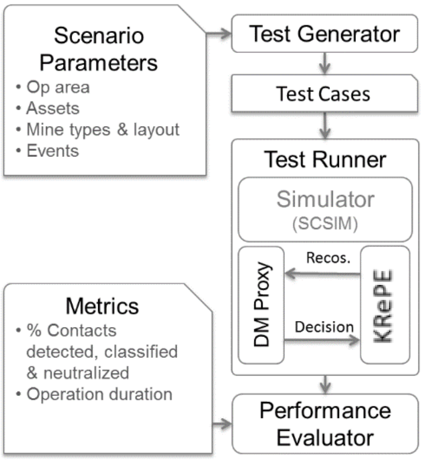


Fig. 2. KRePE simulation driven evaluation

5.2 Experiment Setup

Our experiments used three decision makers and ten randomly generated test scenarios. The first decision maker, “KRePE DM”, confirms the correct root cause with the highest disruption likelihoods and selects a new schedule from those generated to activate. The second decision maker, “Random DM”, randomly chooses a root cause and selects a new schedule from that root cause. The third decision maker, our baseline, “Ignore DM”, ignores KRePE’s recommendations, never changing its schedule when prompted. Comparing performance of these three decision makers allows us to measure the efficacy and correctness of schedules generated by case-base disruption monitoring system.

The performance of each decision maker was evaluated in each of ten randomly generated scenarios, generated. (See Table 2). Scenarios differ primarily in the thirty random events that occur and the positions of mines and mine-like objects. Each event was additionally parameterized with a trigger time (chosen randomly over the first six-hundred hours of the mission) and target unit (chosen randomly among the six tasked assets). The times were chosen in this fashion because events that occur when a unit has already performed all its tasks cause no problems, and therefore are uninteresting to our study. Four mine lines, each with a mine count between ten and thirty, at various depths and mine types were placed randomly in each scenario.

The fixed parameters used in all scenarios included the area searched, and seven assets, consisting of four helicopters, two MCM ships, and one support ship that could assist in tasks if necessary. Each ship and helicopter has available equipment for hunting mines, contact sweeping, detection, and mine neutralization.

5.3 KRePE Metrics

We evaluated KRePE DM, Random DM, and Ignore DM using the following three metrics: (1) *percent contacts detected:* This measures the percentage of mines detected by a unit; (2) *percent mines neutralized:* Percentage all mines are neutralized by a unit and (3) *operation duration:* Total simulation time required to complete the operation.

The first two metrics are calculated based on the true number of mines and mine-like objects generated in the scenario. These summarize the plan's effectiveness in terms of how well the MCM mission goal of searching for and eliminating mines was achieved. Each scenario generated includes a large number of non-mine mine-like objects uniformly spread throughout the threat area, so the percent contacts detected value is an approximation of the percent clearance, or probability that a mine would be detected at any given location. The third metric, operation duration, illustrates a plan's efficiency by measuring the total simulation time required to complete all tasks.

5.4 KRePE Results

Experiments were run on an i7 processor laptop, taking one hour to complete. Figure 3 shows a scatter plot that displays the percentage of existing contacts that were classified correctly and duration of each mission operation measured in simulation hours. The duration of an operation performed by Ignore DM varies little, as the original schedule is never updated, whereas the duration of KRePE DM and Random DM missions can vary greatly. A schedule can be lengthened dramatically when new mine types have been discovered; to ensure safety, many new hunt and/or sweep tasks must be introduced to clear the additional mines. Similarly, if vehicles are damaged beyond repair, the diminished resources can greatly increase mission length. The increased time and repaired schedules allow KRePE DM to outperform Ignore DM by classifying between 95 and 100% of the mine like objects in every mission. Random DM, like KRePE DM, responds to disruptions, but because it does not choose the most likely cause, its task performance is not as high as KRePE DM's. Note that neither Ignore DM nor Random DM represents any real human decision maker; rather these results should be interpreted to show the difficulty of the task and that CDMA's suggestions are benefitting mission performance.

Table 2 shows one-tailed t-test with paired examples. The results include the average and standard deviation for each metric and decision maker. Note: indicate the (small) likelihood that Ignore DM might on average achieve higher values than KRePE DM if many more experiments were undertaken.

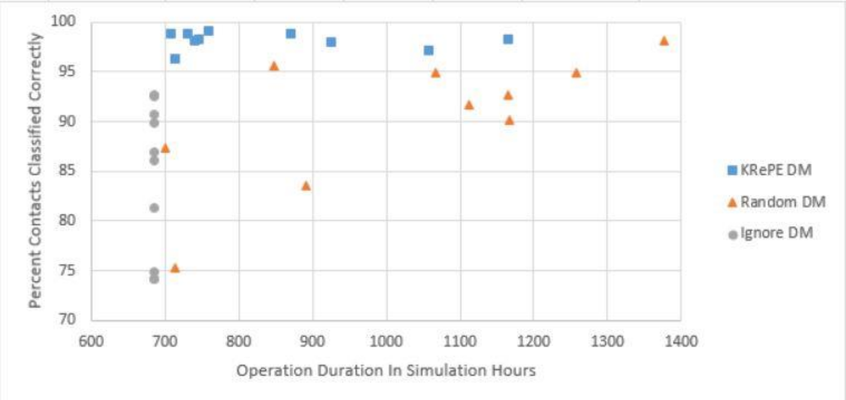
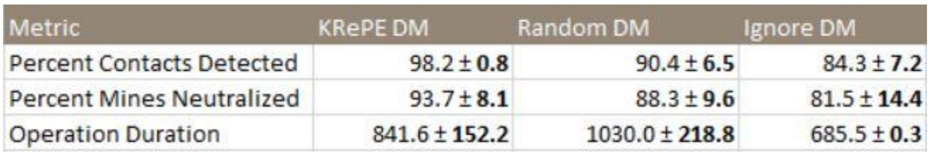


Fig. 3. Scatter Plot of Operation Duration to Percent Contacts Classified Correctly

Table 2. KRePE Results



Metric	KRePE DM	Random DM	Ignore DM
Percent Contacts Detected	98.2 ± **0.8**	90.4 ± **6.5**	84.3 ± **7.2**
Percent Mines Neutralized	93.7 ± **8.1**	88.3 ± **9.6**	81.5 ± **14.4**
Operation Duration	841.6 ± **152.2**	1030.0 ± **218.8**	685.5 ± **0.3**

6 Related Work

Case-based reasoning [1] is a problem solving process based on the adaptation and application of known solutions to new problems. It has been applied to many different domains and problems besides disruption detection.

DISCOVERHISTORY [9] looks for explanations of observations through abductive reasoning, where it maps an observation to a hypothesis that accounts for the observation. DISCOVERHISTORY has been shown to be effective over a large problem space, but is slow with determining disruptions. This is not sufficient for quick detection of immediate issues required by mine countermeasures operations.

A case-based reasoning system, CHEF [7] creates food recipes and explains its own failures. The system tries strategies to see which one can be used to fix the recipe plan. CHEF uses causal rules to explain why its own plan fails. However, the system does not handle constrained resources present in a typical scheduling problem.

The system described in [3] is a CBR system that focuses on wartime equipment maintenance by analyzing feature sets of equipment for maintenance. The system automates the process of deciding the quality of the equipment. CDMA, in contrast, supports a “man-in-the-loop” in order to allow operators to have control over what should be done about disruptions.

7 Conclusion

We presented the CDMA algorithm within the KRePE system that supports monitoring for disruptions and disruption analysis in mine countermeasures operations. Scheduling in this domain is challenging due to the complexities resulting from a large number of tasks that must be allocated over numerous resources. CDMA includes components that assist operation planners by constantly monitoring the environment for changes and providing analysis of discrepancies. Once disruption detection occurred CDMA made it possible for the CLOSR algorithm to reschedule without the need to replan by recommending alternative schedules. We introduced the requirement of minimally disruptive repair as a key operational requirement for automatic schedule repair algorithms in MCM applications.

Our results indicate the efficacy of a case-based strategy; schedule repair was rapid, and created new schedules on demand that ensured the elimination of all mines and increased clearance to a reasonable level. This presents a novel and measurable increase in automated MCM rescheduling capabilities. In the future, we want to apply our system to Unmanned Combat Logistic missions in order to demonstrate effective case-base disruption monitoring with other domains.

8 Bibliographic References

1. Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, *7*(1), 39-59.
2. Boyd, John, R. (1995). The essence of winning and losing. 28 June 1995.
3. Cai, Jiwei., Jia, Yunxian., Gu, Chuang., and Wu, W. (2011). Research of Wartime Equipment Maintenance Intelligent Decision-making Based on Case-Based Reasoning. In *Procedia Engineering* (Volume 15, 2011, pp. 163-167. CEIS 2011).
4. Chief of Naval Operations, Commandant of the Marine Corps, & Commandant of the Coast Guard. (2007). *A Cooperative Strategy for 21st Century Seapower*.
5. Cummings, Mary, and Collins, Angelo. (2010). Autonomous Aerial Cargo/Utility. In *Concept of Operations, Department of the Navy, ONR, Science & Technology*.
6. Garcia, G. A., & Wettergren, T. A. (2012). Future planning and evaluation for automated adaptive minehunting: A roadmap for mine countermeasures theory modernization. In *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics.
7. Hammond, Kristian J. (1986). CHEF: A Model of Case-Based Planning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*. Philadelphia, Pennsylvania.
8. *IRIS. Program documentation. IRIS Reasoner. Vers. 0.6. N.p., 3 Apr. 2008*. Web. 1 Aug. 2015. <http://www.iris-reasoner.org/pages/user_guide.pdf>.
9. Molineaux, M., Kuter, U. and Klenk, M. (2012). DiscoverHistory: Understanding the past in planning and execution. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems* (pp. 989–996. ACM Press, Valencia).
10. Molineaux, M., Auslander, B., Moore, P. G., & Gupta, K. M. (2015). Minimally disruptive schedule repair for MCM missions. In *SPIE Defense+ Security*. International Society for Optics and Photonics.

A CBR Approach to the Angry Birds Game

Adil Paul and Eyke Hüllermeier

Department of Computer Science
University of Paderborn, Germany
`{adil.paul,eyke}@upb.de`

Abstract. In this paper, we present a CBR approach for implementing an agent playing the well-known Angry Birds game. We adopt a preference-based procedure for constructing the case base, collecting experience from a random agent that continually explores the problem-solution space and improves the quality of already found solutions. As the retrieve phase involves finding a game scene similar to a given one, we develop a measure to assess the dissimilarity between two game scenes, which is based on solving appropriate linear assignment problems. A comparison of our agent with state-of-the-art computer programs shows promising results.

1 Introduction

Angry Birds is a popular video game, in which the player has to shoot birds from a slingshot at pigs that are protected with objects from different types of materials, including wood, stone, and ice. Some birds have specific capabilities that allow them to explode, split into several birds, pick up speed, etc. The game has different levels, each level coming with its specific representation of pigs and objects hiding them. A level is solved when all the pigs are destroyed, and the goal of a player is to solve all the levels, keeping the number of shot birds as low as possible.

Since the first edition of the Angry Birds AI competition in 2012, different approaches, ranging from qualitative representation and reasoning over simulation of game scenes to classical supervised machine learning algorithms, have been leveraged to build agents playing the game. In this paper, we develop an Angry Birds agent on the basis of the case-based reasoning (CBR) paradigm. To the best of our knowledge, this is the first CBR approach to Angry Birds. One of the main components of our Angry Birds agent is a case base that stores problem-solution pairs, i.e., game scenes and appropriate best shots. We use a preference-based approach to build the case base, which compares different solutions for a given problem and maintains the better one.

The rest of the paper is organized as follows. In the next section, we briefly review some of the existing approaches for agents playing the Angry Birds game. In Section 3, we present our approach, and in Section 4, we analyze its performance experimentally. We conclude our work and outline possible directions for future work in Section 5.

In Proceedings of the ICCBR 2015 Workshops. Frankfurt, Germany.

2 Existing Approaches

Most of the work so far has been concerned with the representation of the different types of objects in Angry Birds. Lin et al. [7] classify the objects into dynamic, which are mainly convex polygons, and static ones, which comprise concave polygons, and use bounding convex polygons (BCPs) to represent the former and edge detection and Hough transform to detect the latter. Zhang and Renz [12, 13] also make use of the spatial representation of objects and, moreover, reason about their stability. They build on an extension of the rectangle algebra to assess the stability of blocks of objects, upon which they can decide where to hit a block so as to affect it maximally.

In [11], the authors assign a numerical score to each reachable object, based on its physical properties. The score is supposed to reflect the extent of damage it suffers if being hit, and shoots at objects with low stability but high influence on pigs or shelters of pigs. Ferreira et al. [3] also assign a utility value to the objects based on spatial properties, but because of the lack of certainty in the position of the objects, they incorporate concepts of probability and uncertainty to determine the chance of a bird to hit a given target.

Simulation-based approaches include the work by Polceanu and Buche [9], who build their decision making based on the theory of mental simulation. More precisely, their agent observes the effects of performing multiple simulations of different shots in a given game scene and selects the optimal solution based on these results.

The remaining category of approaches encompasses agents that leverage different machine learning algorithms. In order to learn how to judge shots, Narayan-Chen et al. [8] train a weighted majority and a Naive Bayes algorithm on a data set consisting of good and bad shots in different states of the game. Tziortziotis and Buche [10] use a tree structure to represent the objects in a game scene, and formulate the problem of selecting an object for shooting as a regression problem. They associate with each pair of object material and bird a Bayesian linear regression model, building a competitive ensemble of models, whose parameters are estimated in an online fashion. The decision is then made according to the best prediction of the ensemble model.

3 A Case-based Angry Birds Agent

We employ the CBR approach [1] to build an agent that plays the Angry Birds game. The experience-oriented learning and reasoning paradigm of CBR first of all requires the creation of a case base that stores problem-solution pairs. As the problem space in the domain of Angry Birds is infinite, and no exact characterization of an optimal solution (the best shot) for a problem (a description of a game scene) exists, a way of gathering expressive pairs of problems and approximate solutions (game scenes together with reasonably good shots) is needed. Further, a game scene in Angry Birds comprises objects with different shapes, which should be represented and stored appropriately. Thus, a representation that reflects the spatial properties of the different objects involved in the game is another concern. Lastly, once the case base is built and appropriately stored, the problem of retrieving cases similar to a given query case needs to be addressed, which in turn necessitates assessing the similarity between two game scenes. In the following, we elaborate on each of these issues.

3.1 Case Base Construction

The core of a CBR system is a case base that stores previously encountered problems and associated solutions. In the context of Angry Birds, a single case should enclose a problem description part, with a representation of a game scene, covering the slingshot and all objects and pigs, and a solution part, containing the best shot one can execute in the given scene. The notion of an optimal solution in a given game scene, i.e., the shot that will lead to the highest change in score, is actually not well-defined. Therefore, we need a procedure to find solutions of at least close-to-optimal quality.

Inspired by the general framework of preference-based CBR [5], we construct a case base by comparing the quality of solutions that have been tried so far. The basic principle of the approach consists of randomly trying different solutions for a problem and maintaining the best one. The advantages of this approach are two-fold. First, because of its self-adaptive nature, it does not rely on any external domain expert to provide solutions for the potentially infinite number of problems. Second, as the problem and solution space are explored more and more, the extent of the case base is enlarged and its quality is improved over time.

In the context of Angry Birds, we concretise the approach as follows. We let arbitrary agents play in different game scenes and record the game scene along with the shot executed by the agent and the change in score. Once we encounter a game scene which is similar to another one already contained in the case base, and where the agent performs better, we replace the solution part of the old case (i.e., the shot) with the new one. The steps of the process of case base construction are outlined in Figure 1 as a flowchart diagram.

3.2 Case Representation

The Angry Birds game involves different types of objects: a sling, hills, pigs, blocks of stone, wood or ice, TNTs and birds with different capabilities expressed in terms of colours, including red, yellow, blue, black, and white. The Angry Birds Basic Game Playing Software [4] provides two possibilities of representing theses objects: the Minimum Bounding Rectangle (MBR) and the real shape representation. While the MBR segmentation of an object consists solely of finding a rectangle with minimal area, which completely covers it, the real shape segmentation represents the objects more precisely using circles, rectangles and polygons, and distinguishes between hollow and solid objects. As such, the latter is more precise but also more costly to compute. In this paper, we confine ourselves to the MBR representation of objects.

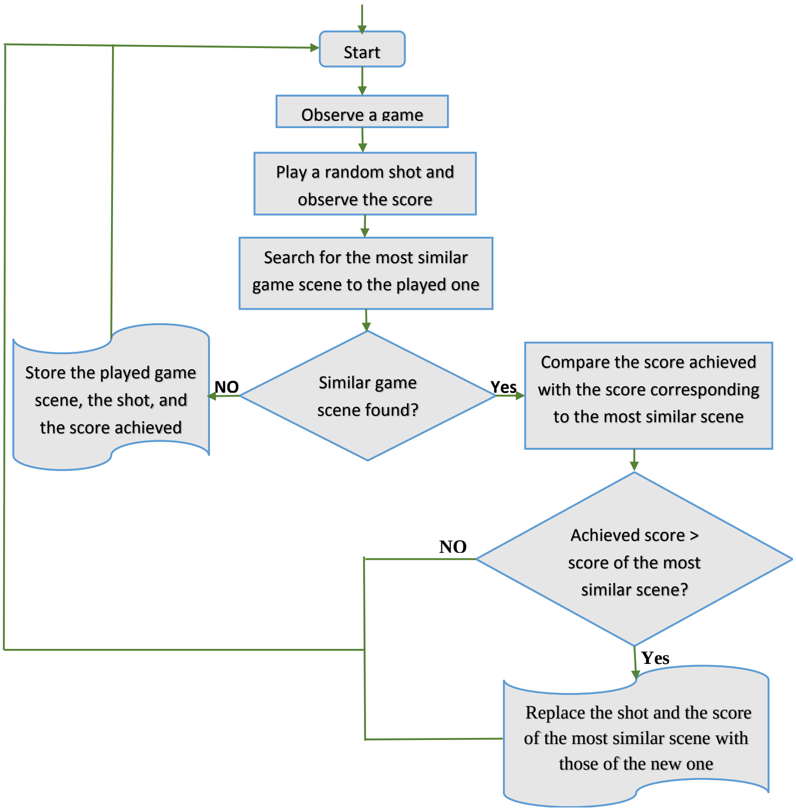

Start
Observe a game
Play a random shot and observe the score
Search for the most similar game scene to the played one
Similar game scene found?
NO
Yes
Store the played game scene, the shot, and the score achieved
Compare the score achieved with the score corresponding to the most similar scene
Achieved score > score of the most similar scene?
NO
Yes
Replace the shot and the score of the most similar scene with those of the new one

Fig. 1. The steps of the case base construction process.

For describing the rectangles, we adopt the interval-based representation, where a rectangle in the 2-dimensional space $\mathbb{R}^2$ has the following form: $R = [l, u] = [l_1, u_1] \times [l_2, u_2]$, where $l = (l_1, l_2)$ and $u = (u_1, u_2)$ are the coordinates of the lower left and upper right vertex of R, respectively. A complete game scene is represented through the set of the MBRs of all objects, together with their type when an object and colour when a bird.

Besides the game scene, collecting the cases also involves recording shots, which constitute the solution part of a case. In the Angry Birds Basic Game Playing Software, a shot is represented in the form of a 6-dimensional vector $s = (x, y, d_x, d_y, t_{shot}, t_{tap})$, where (x, y) and $(x + d_x, y + d_y)$ are the coordinates of the focus and release point, respectively, t_{shot} specifies the releasing and t_{tap} the tapping time of the bird in milliseconds.

To illustrate how a case is constructed, we consider the situation shown in Figure 2. The start game scene is shown in the picture on the left. The resulting scene after performing the shot with the trajectory indicated by the red line is shown in the picture on the right, where the change in score is seen as well.

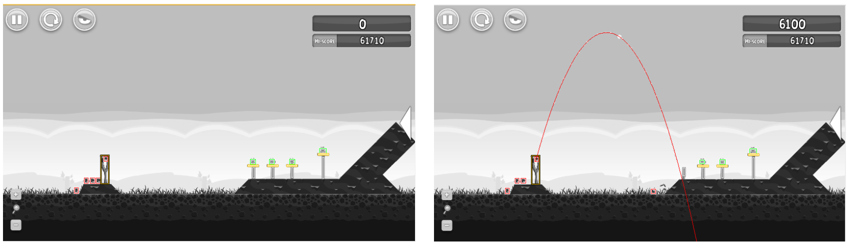


Fig. 2. The game scene before (left) and after (right) performing the shot indicated by the red line in the figure on the right. The MBRs of all objects in both scenes are marked. The change in score after performing the shot is shown on the top right of the figure on the right.

The case extracted from this scenario will contain the original scene, the performed shot and the achieved score, which we represent as follows:

Sling: $l_1 = 200,\ \ u_1 = 305,\ \ l_2 = 216,\ \ u_2 = 363.$
BirdType: *RedBird*.
Hills:
Hill 1 : $l_1 = 471,\ \ u_1 = 237,\ \ l_2 = 839,\ \ u_2 = 384.$
Pigs:
Pig 1 : $l_1 = 645,\ \ u_1 = 290,\ \ l_2 = 659,\ \ u_2 = 300.$
Pig 2 : $l_1 = 504,\ \ u_1 = 314,\ \ l_2 = 514,\ \ u_2 = 321.$
Pig 3 : $l_1 = 543,\ \ u_1 = 313,\ \ l_2 = 353,\ \ u_2 = 323.$
Pig 4 : $l_1 = 584,\ \ u_1 = 313,\ \ l_2 = 595,\ \ u_2 = 323.$
TNTs: -
Blocks:
Block 1: $l_1 = 651,\ \ u_1 = 309,\ \ l_2 = 654,\ \ u_2 = 352.$
Block 2: $l_1 = 509,\ \ u_1 = 330,\ \ l_2 = 513,\ \ u_2 = 351.$
Block 3: $l_1 = 548,\ \ u_1 = 330,\ \ l_2 = 552,\ \ u_2 = 351.$
Block 4: $l_1 = 588,\ \ u_1 = 330,\ \ l_2 = 591,\ \ u_2 = 350.$
Block 5: $l_1 = 643,\ \ u_1 = 302,\ \ l_2 = 663,\ \ u_2 = 304.$
Block 6: $l_1 = 500,\ \ u_1 = 325,\ \ l_2 = 520,\ \ u_2 = 327.$
Block 7: $l_1 = 540,\ \ u_1 = 325,\ \ l_2 = 560,\ \ u_2 = 327.$
Block 8: $l_1 = 579,\ \ u_1 = 325,\ \ l_2 = 599,\ \ u_2 = 327.$
Shot: $x = 208,\ \ y = 315,\ \ d_x = 35,\ \ d_y = 868,\ \ t_{shot} = 0,\ \ t_{tap} = 0.$
Score: 6100.

3.3 Case Retrieval

When the agent is playing, it gets a representation of the current game scene, searches the case base for the case with the most similar game scene and adopts its shot. Therefore, an appropriate measure to assess the similarity respectively dissimilarity between two game scenes is a key prerequisite for a successful agent. We compute the overall dissimilarity between two game scenes as the sum of the dissimilarities between their individual components:

$$\begin{aligned} diss(scene_1, scene_2) = {} & diss(scene_1.Sling, scene_2.Sling) \\ & + diss(scene_1.BirdType, scene_2.BirdType) \\ & + diss(scene_1.Hills, scene_2.Hills \\ & + diss(scene_1.Pigs, scene_2.Pigs) \\ & + diss(scene_1.TNTs, scene_2.TNTs) \\ & + diss(scene_1.Blocks^S, scene_2.Blocks^S) \\ & + diss(scene_1.Blocks^W, scene_2.Blocks^W) \\ & + diss(scene_1.Blocks^I, scene_2.Blocks^I) \ , \end{aligned}$$

where $Blocks^S, Blocks^W$ and $Blocks^I$ denote blocks of stone, wood and ice, respectively.

The dissimilarity of two slings is just the dissimilarity between their MBRs. For the bird type, we compute the dissimilarity as follows:

$$diss(scene_1.BirdType, scene_2.BirdType) = \begin{cases} 0, & \text{if the types are equal,} \\ constant, & \text{otherwise.} \end{cases}$$

Measuring the dissimilarity between two game scenes in each of the remaining components (hills, pigs, TNTs, and blocks) reduces to measuring the dissimilarity between the two sets of rectangles, with potentially different cardinality, corresponding to the MBRs surrounding them. This requires building pairs from the elements of the two sets, between which the dissimilarity is to be computed. The overall dissimilarity between the two sets is then the sum of the dissimilarities between all pairs. We formulate the task of computing the dissimilarity between two sets of rectangles as a (potentially unbalanced) linear assignment problem, where the agents are the elements of one set, tasks are the elements of the other set and the total cost of an assignment is the overall sum of the dissimilarities between all built pairs.

In the following, we proceed with the description of the measure we use for assessing the dissimilarity between two rectangles, prior to detailing our approach to computing the dissimilarity between two game scenes in the above-mentioned components through solving appropriate assignment problems.

Dissimilarity Between Two Rectangles. Different measures exists to assess the dissimilarity between two rectangles in a p-dimensional space. We use the vertex-type distance d_v [2], which is defined for two 2-dimensional rectangles $R_1 = \left[l^{(1)}, u^{(1)}\right] = \left[l_1^{(1)}, u_1^{(1)}\right] \times \left[l_2^{(1)}, u_2^{(1)}\right]$ and $R_2 = \left[l^{(2)}, u^{(2)}\right] = \left[l_1^{(2)}, u_1^{(2)}\right] \times \left[l_2^{(2)}, u_2^{(2)}\right]$, as follows:

$$d_v\left(R_1, R_2\right) = \left(l_1^{(1)} - l_1^{(2)}\right)^2 + \left(u_1^{(1)} - u_1^{(2)}\right)^2 + \left(l_2^{(1)} - l_2^{(2)}\right)^2 + \left(u_2^{(1)} - u_2^{(2)}\right)^2 \ .$$

Dissimilarity Between Two Sets of Rectangles. As stated above, we build on solving an assignment problem to compute the dissimilarity between two sets of rectangles, which represent the MBRs of objects of specific material in two game scenes to be compared.

The linear assignment problem consists of mutually assigning objects of two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ in a cost-optimal manner. Formally, assignment costs are defined in terms of a matrix $C = (c_{ij})$, where c_{ij} denotes the cost of assigning a_i to b_j (and vice versa), $i.j \in [N] = \{1, \ldots, N\}$. The goal, then, is to find an assignment that minimizes the total cost

$$\sum_{i \in [N]} \sum_{j \in [N]} c_{ij} x_{ij}$$

with

$$x_{ij} = \begin{cases} 1, & \text{if } a_i \text{ and } b_j \text{ are mutually assigned,} \\ 0, & \text{otherwise.} \end{cases} ,$$

subject to the following constraints:

$$\sum_{j \in [N]} x_{ij} = 1 \text{ for all } i \in [N],$$

$$\sum_{i \in [N]} x_{ij} = 1 \text{ for all } j \in [N],$$

The Hungarian algorithm [6] is one of the best-known methods for solving the assignment problem. It is mainly based on the observation that adding or subtracting a constant from all the entries of a row or a column of the cost matrix does not change the optimal solution of the underlying assignment problem. Thus, the algorithm proceeds iteratively, subtracting and adding constants in each step to specific rows and columns of the cost matrix, in such a way that more and more zero-cost pairs are built, until an optimal solution can be found. We refer to [6] for a detailed description of the Hungarian algorithm.

In the simplest form of the assignment problem, the number of objects in A and B are equal. For the problem at hand, this assumption does not hold; instead, we are dealing with an unbalanced assignment problem. To handle such problems, one usually introduces dummy rows or columns in the cost matrix, depending on which number exceeds the other. Normally, the introduced entries are filled with zeros, but this does not fit our purpose, because the addition or removal of objects will normally influence the best shot in a scene. We overcome this issue by associating a penalty with objects that remain unassigned. The penalty term for an unassigned rectangle is its distance to the zero-perimeter rectangle located at the origin, i.e., $R = [0,0] \times [0,0]$.

4 Experimental Results

We begin our experimental analysis with the construction of the case base, in which we proceed as follows. We run a random agent that chooses the coordinates of the shot to be executed fully at random, and we restrict ourself to the first 21 levels of the “Poached Eggs” episode of Angry Birds. The agent plays each level several times and the cases from each level are first collected in separate files. The distribution of the number of cases we gathered over the different levels of the game, shown in Table 1, was not uniform. That is, we dedicate more examples to harder levels than to easier ones. At the end, we combine all cases in one file, ending up with a case base of total size of $11,703$, which serves as the main case base for our agent.

Table 1. The number of cases we collected in each of the 21 levels of the game.

Level	# cases
1	50
2	50
3	50
4	130
5	50
6	100
7	50

Level	# cases
8	50
9	100
10	647
11	50
12	50
13	182
14	100

Level	# cases
15	50
16	50
17	50
18	400
19	200
20	100
21	100

After the case base was constructed, we first tested the performance of our agent on the above-mentioned levels. To this end, we let the agent play 10 games and report the minimal, maximal, and average score for each level, together with the standard deviation, in Table 2.

To get an idea of how our agent performs in comparison to others, Figure 3 plots the average score of our agent from Table 2 together with the scores of the naive agent, the top-3 agents of the 2013 and 2014 participants of the AI competition, and the average scores of all 30 participants, on all 21 levels, based on the 2014 benchmarks provided on the *aibirds.org* website. This comparison shows that our agent clearly outperforms both the naive and the average agent in both per-level and total scores, and is even competitive to the top-3 agents.

5 Conclusion and Future Work

We made use of CBR to build an Angry Birds playing agent. The results of an experimental study, in which we compared our agent with others, including the top-3 systems of previous AI competitions, are very promising, especially in light of the rather simple implementation of our agent so far. In fact, we are convinced that our agent’s performance can be further enhanced through the collection of more cases and the refinement of the different steps of the CBR cycle.

More concretely, this work can be extended along the following directions. First, the real shape instead of the MBR representation can be used to represent the objects involved in the game. Second, a weighted version of the distance measure between game scenes can be learnt. Third, cases from levels of the

Table 2. The minimal, maximal, and average score, and the standard deviation of our agent in 10 games on the first 21 levels of the “Poached Eggs” episode of Angry Birds.

Level	Min. score	Max. score	Mean score	Standard deviation
1	28950	30790	29735	704.955
2	60950	61520	61293	188.388
3	42510	42540	42529	11.005
4	10660	36810	22500	9174.102
5	59680	67760	65301	2302.744
6	18020	35620	32096	6115.800
7	31180	46200	42486	5777.303
8	54110	54120	54111	3.162
9	32130	50020	44525	5874.565
10	32650	59920	46980	9294.536
11	54130	57390	55634	910.668
12	53010	54880	54248	550.713
13	21530	48090	33036	8987.933
14	49250	73760	65553	6858.706
15	37760	48540	46486	3166.492
16	54410	64790	61646	3073.714
17	46290	49900	48492	1224.444
18	39710	60830	49888	7150.137
19	31710	38550	33127	1999.445
20	34030	59140	46527	10113.806
21	59720	96880	70332	11020.633

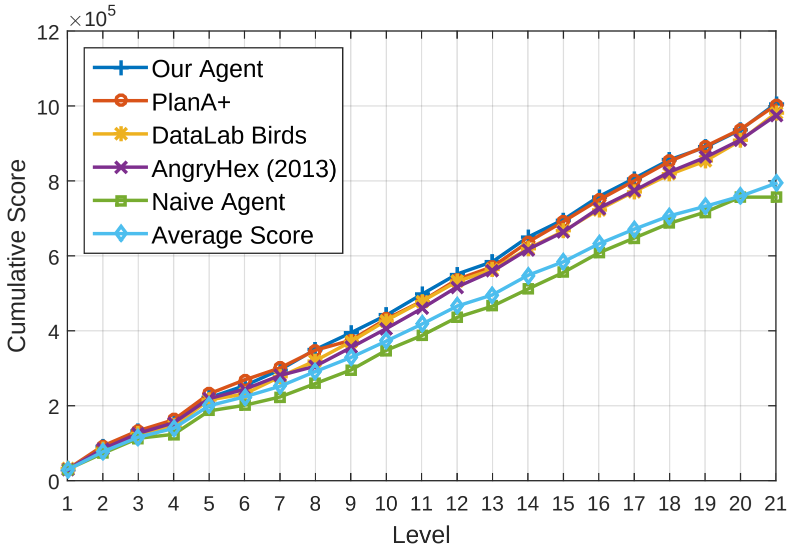


Fig. 3. The cumulative scores of our agent, the top 3 agents of the 2013 and 2014 participants of the AI competition, the naive agent, and the average agent, on the first 21 levels of the “Poached Eggs” episode of Angry Birds.

game other than the ones of the “Poached Eggs” episode can be extracted to increase the size and coverage of the case base. Fourth, since our agent does not realize any adaptation of the retrieved solutions so far, a sophisticated adaptation strategy could be another means to improve performance.

Acknowledgments. This work has been supported by the German Research Foundation (DFG).

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications* 7(1), 3959 (1994)
2. Bock, H.H. Analysis of symbolic data: Exploratory methods for extracting statistical information from complex data. E. Diday (Ed.). *Springer-Verlag New York*, Inc., Secaucus, NJ, USA (2000)
3. Ferreira, L.A., Lopes, G.A.W., Santos, P.E.: Combining qualitative spatial representation utility function and decision making under uncertainty on the Angry Birds domain. In: *IJCAI 2013 Symposium on AI in Angry Birds* (2013)
4. Ge X., Gould, S., Renz, J., Abeyasinghe, S., Keys, J., Wang, A., Zhang, P. Angry Birds basic game playing software, version 1.32. Technical report. Research School of Computer Science, The Australian National University (2014)
5. Hüllermeier, E., Schlegel, P.: Preference-based CBR: First steps toward a methodological framework. In: Ram, A., Wiratunga, N. (eds.) *ICCBR 2011*. LNCS, vol. 6880, pp. 7791. Springer, Heidelberg (2011)
6. Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Research Logistics*, 2: 8397 (1955)
7. Lin, S., Zhang, Q., Zhang, H.: Object representation in Angry Birds game. In: *IJCAI 2013 Symposium on AI in Angry Birds* (2013)
8. Narayan-Chen, A., Xu, L., Shavlik, J. An empirical evaluation of machine learning approaches for Angry Birds. In: *IJCAI 2013 Symposium on AI in Angry Birds* (2013)
9. Polceanu, M., Buche, C.: Towards a theory-of-mind-inspired generic decision-making framework. In: IJCAI 2013 Symposium on AI in Angry Birds (2013)
10. Tziortziotis, N., Papagiannis, G., Blekas, K. A Bayesian ensemble regression framework on the Angry Birds game. In: *ECAI 2014 Symposium on Artificial Intelligence in Angry Birds* (2014)
11. Wałęga, P., Lechowski, T., Zawidzki, M. Qualitative physics in Angry Birds: first results. In: *ECAI 2014 Symposium on Artificial Intelligence in Angry Birds* (2014)
12. Zhang, P., Renz, J. Qualitative spatial representation and reasoning in Angry Birds: first results. In: *IJCAI 2013 Symposium on AI in Angry Birds* (2013)
13. Zhang, P., Renz, J. Qualitative spatial representation and reasoning in Angry Birds: the extended rectangle algebra. In: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning*, KR14, p. to appear, Vienna, Austria (2014)

Flexible Plan-Subplan Matching for Plan Recognition in Case-Based Agents

Keith A. Frazer[1,3], Swaroop S. Vattam[2], and David W. Aha[3]

[1]College of Computing, Georgia Institute of Technology, Atlanta, GA
[2]NRC Postdoctoral Fellow; Naval Research Laboratory (Code 5514); Washington, DC
[3]Navy Center for Applied Research in Artificial Intelligence;
Naval Research Laboratory (Code 5514); Washington, DC
kfrazer3@gatech.edu | {swaroop.vattam.ctr.in, david.aha}@nrl.navy.mil

Abstract. Plan-subplan matching is an important step in case-based plan recognition. We present RelaxedVF2, an algorithm for plan-subplan matching for plans encoded using the Action Sequence Graph representation. RelaxedVF2 is a subgraph monomorphism algorithm that affords flexibility and error tolerance for plan-subplan matching. We present a study comparing RelaxedVF2 with an alternate degree-sequence matcher that we used in our prior work and found that RelaxedVF2 attains higher plan recognition accuracy on a paradigmatic domain.

Keywords: Plan Recognition, Case-Based Reasoning, Action-Sequence Graph, Relaxed Graph Matching, Error-Tolerant

1. Introduction

An agent on a team must cooperate and coordinate its actions with its teammates, requiring the ability to recognize its teammates' plans. *Plan recognition* refers to the task of observing a teammate's current actions, inferring the plan governing those actions, and predicting that teammate's future actions. A plan recognizer takes as input the observed portion of a plan (subplan) and outputs a (predicted) full plan. A case-based plan recognizer matches its input subplan to a set of plans in its case base and retrieves a most similar plan to the given subplan. We assume that the most similar plan best explains the observed subplan. Plan-subplan matching is therefore a key component of case-based plan recognition (CBPR).

The algorithm used for plan-subplan matching depends on the representation of plans. Typically plans are represented as (a sequence of) propositions in first-order predicate logic. We instead use an *Action Sequence Graph (ASG)* representation (Vattam et al., 2014; 2015), which has some nice properties: (1) it captures the topology of the propositional plans using graphs, (2) better lends itself to vectorization and approximate matching, (3) and makes the matching process more robust to input errors (Vattam et al., 2015). ASG represents a plan as

In Proceedings of the ICCBR 2015 Workshops. Frankfurt, Germany.

a labeled directed multigraph. Plan-subplan matching using ASGs reduces to graph-subgraph matching.

We introduce the RelaxedVF2 algorithm for graph-subgraph matching that is tailored to matching plans represented as ASGs. RelaxedVF2 is an extension of the popular VF2 algorithm (Cordella et al., 2004) for subgraph isomorphism. The extensions to VF2 we propose transform it into a subgraph monomorphism matching algorithm, which makes RelaxedVF2 better suited for additional edges that arise between the nodes of states and actions as a plan's observed execution progresses.

In §2 we present related work on CBPR and graph matching techniques. In §3 we present the ASG representation for plans. In §4 we present our plan-subplan matching approach, including the RelaxedVF2 algorithm and the scoring function. In §5 we present an initial empirical study comparing the performance of RelaxedVF2 to an alternative degree-sequence matching algorithm that we used in our prior work (Vattam et al., 2015). Our results show that RelaxedVF2 compares favorably to the alternative approach. We conclude and discuss future research plans in §6.

2. Related Research

Several approaches has been proposed to address the problem of plan recognition (Sukthankar et al., 2014), including consistency-based (e.g., Hong, 2001; Kautz & Allen, 1986; Kumaran, 2007; Lau et al., 2003; Lesh & Etzioni, 1996), and probabilistic approaches (e.g., Bui, 2003; Charniak & Goldman, 1991; 1993; Geib & Goldman, 2009; Goldman et al., 1999; Pynadath & Wellman, 2000). Both types are "model-heavy", requiring accurate models of an actor's possible actions and how they interact to accomplish different goals. Engineering these models is difficult and time consuming. Furthermore, these plan recognizers perform poorly when confronted with novel situations and are brittle when the operating conditions deviate from model parameters.

CBPR is a model-lite, less studied approach to plan recognition. Existing CBPR approaches (e.g., Cox & Kerkez, 2006; Tecuci & Porter, 2009) eschew generalized models for plan libraries that contain plan instances which can be gathered from experience. CBPR algorithms can respond to novel inputs outside the scope of their plan library by using plan adaptation techniques. However, earlier CBPR approaches were not error-tolerant.

In contrast, our work on SET-PR focuses on error-tolerant CBPR (Vattam et al., 2014; 2015). We showed that SET-PR is robust to three kinds of inputs errors (missing, mislabeled, and extraneous actions). One of the factors contributing to its robustness is that SET-PR uses an ASG plan representation and the degree sequence similarity function for plan-subplan matching. Although we previously showed that SET-PR was robust to input errors, there is room for improvement.

VF2 (Cordella et al., 2004) is an exact graph matching algorithm for finding node-induced subgraph isomorphisms. It is one of the few such algorithms applicable to directed multigraphs. Our extension, RelaxedVF2, transforms VF2 from finding node-induced subgraph isomorphisms to subgraph monomorphisms (see Figure 1 for an illustration on how these differ) and by modifying it to return partial mappings from graph to subgraph when no complete match is available.

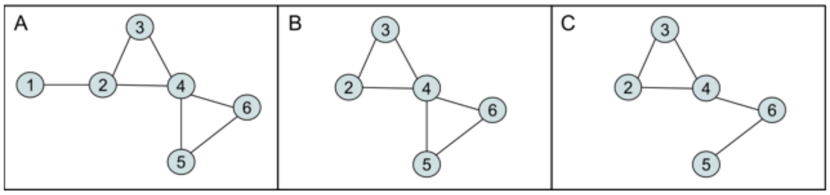


Figure 1: Graph B is a *node-induced isomorphism* of Graph A because it is missing a node (1) but preserves all edges between nodes shared in both Graphs A and B. Graph C is a *monomorphism* of A because it is missing a node (1) and an edge (between 4 and 5). Definitions are provided in Section 4.1.

3. Plan Representation: Action Sequence Graphs

Suppose a plan is modeled as an *action state sequence* $\mathbb{s} = \langle(\boldsymbol{a_0}, \boldsymbol{s_0}), \dots, (\boldsymbol{a_n}, \boldsymbol{s_n})\rangle$, where each action $\boldsymbol{a_i}$ is a ground operator in the planning domain, and $\boldsymbol{s_i}$ is a ground state obtained by executing $\boldsymbol{a_i}$ in $\boldsymbol{s_{i-1}}$, with the caveat that $\boldsymbol{s_0}$ is an initial state, $\boldsymbol{a_0}$ is null, and $\boldsymbol{s_n}$ is a goal state. An action $\boldsymbol{a}$ in $(\boldsymbol{a}, \boldsymbol{s}) \in \mathbb{s}$ is a ground literal $\boldsymbol{p} = p(o_1\colon t_1, \dots, o_n\colon t_n)$, where $p \in \boldsymbol{P}$ (a finite set of predicate symbols), $o_i \in \boldsymbol{O}$ (a finite set of object types), and t_i is an instance of o_i (e.g., stack(block:A, block:B)). A state $\boldsymbol{s}$ in $(\boldsymbol{a}, \boldsymbol{s}) \in \mathbb{s}$ is a set of ground literals (e.g., {on(block:A,block:B), on(block:B,substrate:TABLE)}).

An *Action Sequence Graph (ASG)* is a graphical representation of a plan that preserves its topology (including the order of the propositions and their arguments). Vattam et al. (2014; 2015) provide a detailed definition of ASGs and their generation. An ASG is automatically generated by transforming individual propositions in a plan into predicate encoding graphs, and by taking the union of all the individual predicate encoding graphs so as to maintain the total order of the plan. Figure 2 shows an example proposition and its corresponding predicate encoding graph. Figure 3 shows an example full plan and its corresponding ASG. An ASG is a labeled directed multigraph, which constrains the set of graph matching algorithms that can manipulate them.

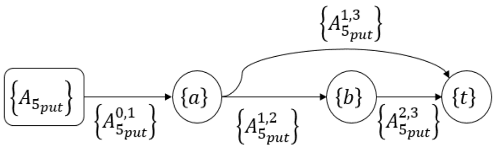


Figure 2: A predicate encoding graph corresponding to $\boldsymbol{p} = put(block\colon a, block\colon b, table\colon t)$

4. Robust Plan-Subplan Matching

As our goal is error-tolerant plan recognition, our approach requires plan-subplan matching that is robust to input errors. Plan-subplan matching requires a measure of similarity. As we encode our plans in the case library and the input subplans as graphs, we utilize maximum common subgraph *monomorphism* as a measure of similarity between them rather than the more conventional maximum common subgraph *isomorphism* measure.

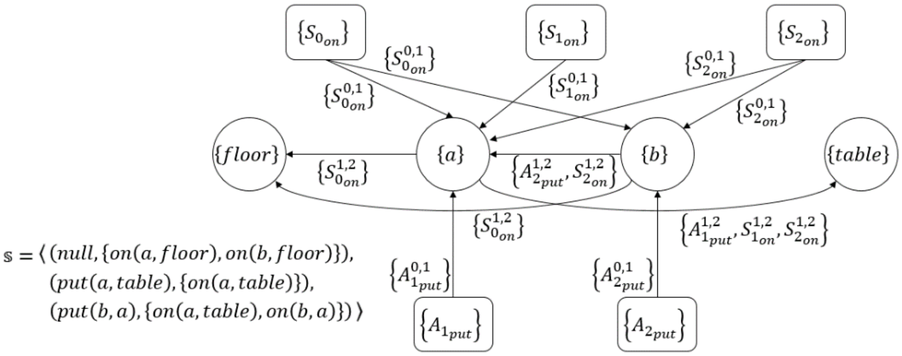


Figure 3: An example of a plan with three action-state sequences and its corresponding ASG

Let G_1, G_2 be graphs composed of sets of vertices and edges V_1, V_2 and E_1, E_2 respectively. G_2 is *isomorphic* to a subgraph of G_1 if there exists a one-to-one mapping between each vertex of V_2 and a vertex in V_1 and the number of edges between nodes in the mapping are maintained. G_2 is instead *monomorphic* if it consists of any subset of the vertices and edges of G_1. Monomorphism must be utilized over isomorphism when matching incomplete subplans to complete plans in the case library because as plans are observed new edges are often added relating existing action and state vertices.

RelaxedVF2 (§4.1), an exact graph matching algorithm, does not return a similarity score. It instead returns a one-to-one mapping of nodes between the subplan and plans in the case library. While the length of the maximum common subgraph is often used to score matches, we instead developed a more nuanced candidate scoring algorithm (§4.2) to increase matching accuracy.

4.1 RelaxedVF2

RelaxedVF2 (Algorithm 1) computes the maximum common subgraph monomorphism between two labeled directed multigraphs. Here we refer to node-induced isomorphism as a subset of the nodes with all corresponding edges between them.

VF2 matches two graphs, G_1 and G_2, using semantic and syntactic feasibility functions to iteratively add compatible nodes of the graphs to an internal mapping, M, which is expressed as a set of pairs (n, m) that represent the mapping of a node $n \in G_1$ with a node $m \in G_2$. Therefore, a mapping M is a *graph* isomorphism if it is a bijective function that preserves the branching structure of the two graphs, and M is a *subgraph* isomorphism if the mapping is an isomorphism between G_2 and a subgraph of G_1.

The original VF2 algorithm uses five syntactic feasibility rules to check if a pair (n, m) can be included in M. These rules are listed in Table 1.

Table 1: Syntactic feasibility rules for VF2 and RelaxedVF2

VF2	**RelaxedVF2**
$R_{pred}(s,n,m) \Leftrightarrow \big(\forall n' \in M_1(s) \cap Pred(G_1,n) \exists m' \in Pred(G_2,m) \mid (n',m') \in M(s)\big) \wedge \big(\forall n' \in M_2(s) \cap Pred(G_2,n) \exists m' \in Pred(G_1,m) \mid (n',m') \in M(s)\big)$	$R_{pred}(s,n,m) \Leftrightarrow \begin{pmatrix} \forall n' \in M_2(s) \cap Pred(G_2,n) \\ \exists m' \in Pred(G_1,m) \mid (n',m') \in M(s) \end{pmatrix}$
$R_{succ}(s,n,m) \Leftrightarrow \big(\forall n' \in M_1(s) \cap Succ(G_1,n) \exists m' \in Succ(G_2,m) \mid (n',m') \in M(s)\big) \wedge \big(\forall n' \in M_2(s) \cap Succ(G_2,n) \exists m' \in Succ(G_1,m) \mid (n',m') \in M(s)\big)$	$R_{succ}(s,n,m) \Leftrightarrow \begin{pmatrix} \forall n' \in M_2(s) \cap Succ(G_2,n) \\ \exists m' \in Succ(G_1,m) \mid (n',m') \in M(s) \end{pmatrix}$
$R_{in}(s,n,m) \Leftrightarrow (Card(Succ(G\_1,n) \cap T\_1\hat{}in\ (s)) \geq Card(Succ(G\_2,n) \cap T\_2\hat{}in\ (s))) \wedge (Card(Pred(G\_1,n) \cap T\_1\hat{}in\ (s)) \geq Card(Pred(G\_2,n) \cap T\_2\hat{}in\ (s)))$	$R_{in}(s,n,m) \Leftrightarrow (Card(Succ(G\_1,n) \cap T\_1\hat{}in\ (s)) \geq Card(Succ(G\_2,n) \cap T\_2\hat{}in\ (s))) \wedge (Card(Pred(G\_1,n) \cap T\_1\hat{}in\ (s)) \geq Card(Pred(G\_2,n) \cap T\_2\hat{}in\ (s)))$
$R_{out}(s,n,m) \Leftrightarrow (Card(Succ(G\_1,n) \cap T\_1\hat{}out\ (s)) \geq Card(Succ(G\_2,n) \cap T\_2\hat{}out\ (s))) \wedge (Card(Pred(G\_1,n) \cap T\_1\hat{}out\ (s)) \geq Card(Pred(G\_2,n) \cap T\_2\hat{}out\ (s)))$	$R_{out}(s,n,m) \Leftrightarrow (Card(Succ(G\_1,n) \cap T\_1\hat{}out\ (s)) \geq Card(Succ(G\_2,n) \cap T\_2\hat{}out\ (s))) \wedge (Card(Pred(G\_1,n) \cap T\_1\hat{}out\ (s)) \geq Card(Pred(G\_2,n) \cap T\_2\hat{}out\ (s)))$
$R_{out}(s,n,m) \Leftrightarrow (Card(\tilde{N}\_1\ (s) \cap Pred(G\_1,n)) \geq Card(\tilde{N}\_2\ (s) \cap Pred(G\_2,n))) \wedge (Card(\tilde{N}\_1\ (s) \cap Succ(G\_1,n)) \geq Card(\tilde{N}\_2\ (s) \cap Succ(G\_2,n)))$	

The first two rules determine match compatibility based on an equivalent number of incoming and outgoing edges per node. The final three rules look ahead to adjacent nodes to prune the search tree. We adapted VF2 to relax its enforcement of graph/subgraph edge counts while maintaining rules disqualifying additional edges in the subgraph not present in the graph. We removed the fifth rule because strict lookahead rules run counter to our goal of increased error tolerance. We also made modifications to enable returning partial matches, removing the rule that matches must be equal in length to the subgraph.

We optimized RelaxedVF2 for graph recognition, and thus primarily rely on the semantic similarity of node labels to restrict our search space. Our simple semantic feasibility function uses an exact string match of the node labels. Any plan recognizer using this algorithm would need to provide as input its own domain-specific semantic similarity measure. RelaxedVF2 uses a depth-first search through all possible nodes that can be added based on semantic and structural similarity. It then progresses to nodes matching on only semantic similarity before finally adding nodes matching using only structural similarity.

Algorithm 1: RelaxedVF2

```
PROCEDURE RelaxedVF2($s, G_1, G_2$)
  INPUT:     An intermediate state $s$ (the initial state $s_0$ has $M(s_0) = \emptyset$) and two graphs
  OUTPUT:  the mappings between $G_1$ and $G_2$
  IF $M(s)$ covers all the nodes of $G_2$ THEN
    OUTPUT $M(s)$   //The function $M$ returns the mappings between nodes of $G_1, G_2$ in state $s$
  ELSE
    L = [ ]    //Sorted list of feasible pairs
    mappingFound = False
    $P(s) \leftarrow$ candidate pairs for inclusion in $M(s)$    //Used candidate pairs function from VF2
    FOREACH $(n, m) \in P(s)$
      IF $F(s, n, m)$ THEN
        L ← L $\cup$ $(n, m)$
    WHILE NOT mappingFound    //Loop until match is found or no more candidates
      $s' \leftarrow M(s) \cup$ L.pop$(n, m)$    //Get the top feasible pair from list
      mappingFound = RelaxedVF2($s', G_1, G_2$)    //Recursive call
    IF NOT mappingFound    //Output partial match if no match found
      OUTPUT $M(s)$
    Restore data structures
```

4.2 Scoring

The size of the largest common subgraph can be used as a similarity measure (Bergmann, 2002). VF2 is error tolerant and will return matches even if they are of lower quality. Therefore, we designed a metric that is a function of both match size and quality. The match algorithm scores 1 point for every full match based on both semantics and structure (0.7 per semantic match and 0.3 per structural match, based on previous weights used in SET-PR (Vattam et al., 2015)). After retrieving all matches of the subgraph against the case library this score is then used to sort and find the best match.

5. Empirical Study

In this study, we compare plan-subplan matching using two similarity measures on ASGs: (1) RelaxedVF2, and (2) DSQ (degree-sequence matcher) (Vattam et al., 2014; 2015). Our claim is that RelaxedVF2 offers better performance compared to DSQ.

The default plan representation consists of action-state sequences ($\langle(\boldsymbol{a_0}, \boldsymbol{s_0}), \dots, (\boldsymbol{a_n}, \boldsymbol{s_n})\rangle$). We also evaluated a plan representation consisting of only action sequences ($\langle(\boldsymbol{a_0}), \dots, (\boldsymbol{a_n})\rangle$) because state information is not always available in all planning domains and it presents a more difficult challenge for DSQ. This yields four conditions: RelaxedVF2$_{\text{ActionStates}}$, DSQ$_{\text{ActionStates}}$, RelaxedVF2$_{\text{ActionsOnly}}$, and DSQ$_{\text{ActionsOnly}}$.

We empirically test whether, for error tolerant CBPR, using RelaxedVF2 outperforms DSQ for both the ActionStates and ActionsOnly conditions. We use plan recognition accuracy as our performance metric, where accuracy is defined as the ratio of queries that resulted in correct plan retrieval to the total number of queries.

5.1 Empirical method

We conducted our experiments in the Blocks World domain, which is simple and allows us to quickly generate a plan library L with desired characteristics. We used SHOP2 (Nau et al., 2003) to generate L's plans as follows. We generated 20 random initial states and paired each with 5 randomly generated goal states to obtain 100 planning problems. Each was given as input to SHOP2 to obtain a plan. We fixed plan length to 20 by discarding any whose length was not 20 and generating a new one (with a different goal state) in its place. This distribution was chosen because it is challenging for plan recognition.

We evaluated plan recognition accuracy using a leave-one-in strategy (Aha & Breslow, 1997). For each compared condition:

1. We randomly selected a plan π in L (π is *not* deleted from L).
2. We introduced a fixed percentage of error into π consisting of a uniform distribution of missing, mislabeled, and extraneous actions and random distortions of states associated with those actions. The error levels that we tested were $\{0\%,10\%,20\%,30\%,40\%,50\%\}$.
3. The error-π plan was then used to incrementally query L to retrieve a plan. For example, if error-π had 20 steps, the evaluator performed 11 queries at the following plan lengths: 0% (initial state only, no actions are observed), 10% (first two actions and states are observed), and so on until 100% (full plan is observed).
4. Each query derived from error-π was used to retrieve the top matching plan π^{sol}. If π was equal to π^{sol}, it was considered a success and a failure otherwise.
5. We repeated steps 1-4 for all 100 plans in L in each of 20 trials.

This yields 1100 queries per error percent level per trial, yielding 132,000 queries (1100 queries × 6 error levels × 20 trials). We computed average accuracy over 20 trials.

5.2 Results and discussion

We computed mean accuracy for each percentError (in [0.0,0.5] with increments of 0.1) and each percentAction (in [0.0,1.0] with increments of 0.1) for RelaxedVF2 and DSQ. The results are shown in Figures 4 and 5 for ActionStates and ActionsOnly, respectively. Our results show that for all error levels and percent actions RelaxedVF2's mean accuracy was higher than DSQ's. In ActionStates, RelaxedVF2 achieves 50% accuracy by 20% actions at all error levels, but DSQ only achieves 50% accuracy at 100% actions at only 0% and 10% error. In ActionsOnly, RelaxedVF2 achieves 50% accuracy by 40% actions at all error levels, but DSQ only approaches 50% accuracy at 100% actions with 0% error.

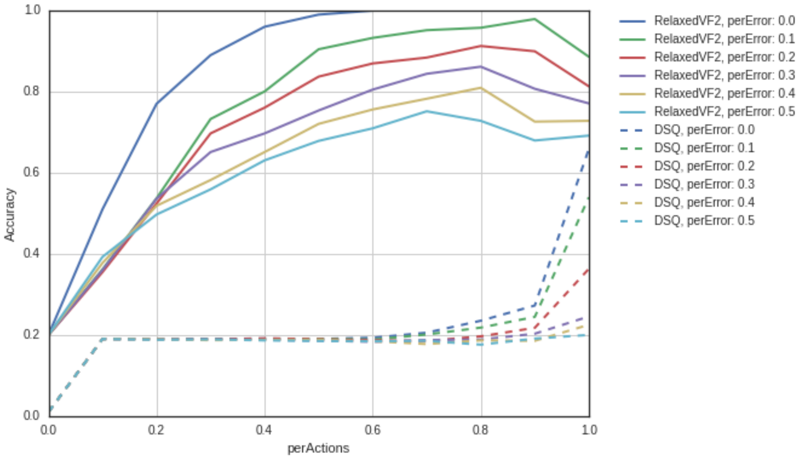


Figure 4: Mean plan recognition accuracy for the ActionStates conditions

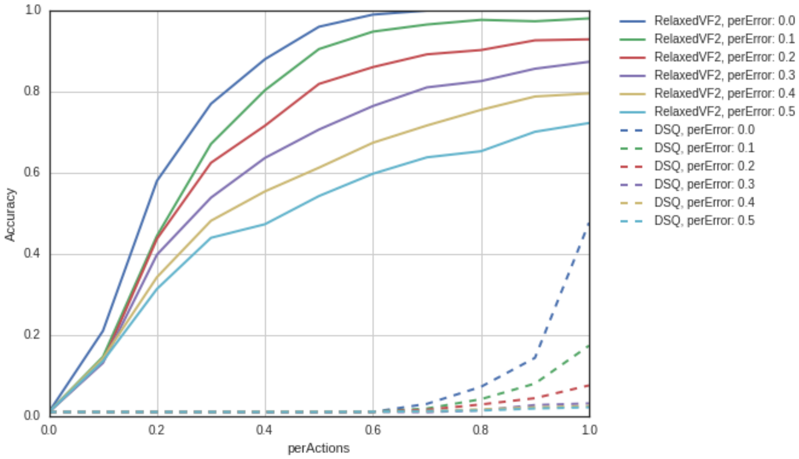


Figure 5: Mean plan recognition accuracy for the ActionsOnly conditions

We conducted a one-way ANOVA test to compare the effects of percent Actions (0%-100%), percent Error (0%-50%) and the matching algorithms (RelaxedVF2, DSQ) on accuracy. There was a significant effect on accuracy at $p < 0.05$ with respect to the matching algorithms (F(1,17)=18687550.204, p=0.0). This analysis shows that RelaxedVF2 significantly outperformed DSQ, which lends support to our claim.

DSQ performs considerably worse without state information because the ASGs become much smaller. The degree sequences across the partitions of the smaller graphs will yield similar values, preventing DSQ from disambiguating the different plans.

Surprisingly, accuracy decreased around 80% actions in RelaxedVF2 in the ActionStates condition (Figure 4). As the error level increases this dip occurs earlier. We hypothesize that the more densely connected graphs resulting from additional action-state information causes these graphs to more closely resemble each other, thus reducing recognition accuracy. We plan to investigate this in future work.

Given that RelaxedVF2 is an exact graph matching algorithm and DSQ is an approximate algorithm, DSQ should have a significantly shorter runtime. In this study, RelaxedVF2 had a mean runtime (in seconds) of 0.121 and 0.045 in the ActionStates and ActionsOnly conditions, respectively. DSQ mean runtime was 0.020 and 0.019 in these conditions. We subjected the mean runtimes to a t-test and found the differences in the runtimes to be significant at $p < 0.05$ for both conditions.

6. Summary

CBPR under imperfect observability requires error tolerant plan-subplan matching, which requires flexible representation and matching algorithms. In earlier work we introduced the ASG representation for plan recognition and degree-sequence plan matching (Vattam et al., 2014; 2015). Although this matching algorithm worked reasonably well, there remained room for improvement. Here we presented RelaxedVF2, an alternative plan-subplan matching algorithm. It is a subgraph monomorphism algorithm, and thus affords flexibility and error tolerance in matching compared to VF2. In our empirical study we found support for our claim that, for error-tolerant CBPR, RelaxedVF2 can outperform the degree-sequence matcher, at least for the paradigmatic domain we studied.

In future work, we will investigate whether the same result occurs when using datasets from additional domains to address the single dataset limitation of our current study. We also plan to integrate RelaxedVF2 into our plan recognition architecture to complement the existing methods. We also plan to do a comparative study with other state-of-the-art plan recognizers.

Acknowledgements

Thanks to OSD ASD (R&E) for sponsoring this research. Swaroop Vattam conducted this research while an NRC post-doctoral research associate at NRL. Keith Frazer conducted this research while an NREIP intern at NRL. The views and opinions in this paper are those of the authors and should not be interpreted as representing the official views of NRL or OSD.

References

Aha, D. W., & Breslow, L. A. (1997). Refining conversational case libraries. *Proceedings of the Second International Conference on CBR* (pp. 267-278). Providence, RI: Springer-Verlag.

Bergmann, R. (2002). *Experience management: Foundations, development methodology, and Internet-based applications*. Berlin, Germany: Springer-Verlag.

Bui, H. (2003). A general model for online probabilistic plan recognition. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (pp. 1309–1315). Acapulco, Mexico: Morgan Kaufmann.

Charniak, E., & Goldman, R. (1991). A probabilistic model of plan recognition. P*roceedings of the Ninth National Conference on Artificial Intelligence* (pp. 160–165). Anaheim, CA: AAAI Press.

Charniak, E., & Goldman, R. (1993). A Bayesian model of plan recognition. *Artificial Intelligence*, *64*, 53–79.

Cordella, L., P., Foggia, P., Sansone, C., & Vento, M. (2004). A (sub)graph isomorphism algorithm for matching large graphs. *Transactions on Pattern Analysis and Machine Intelligence*, *26*(10), 1367-1372.

Cox, M.T., & Kerkez, B. (2006). Case-based plan recognition with novel input. *Control and Intelligent Systems*, *34*(2), 96–104

Geib, C.W., & Goldman, R.P. (2009). A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, *173*(11), 1101–1132.

Goldman, R.P., Geib, C.W., & Miller, C.A. (1999). A new model of plan recognition. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (pp. 245–254). Bled, Slovenia: Morgan Kaufmann.

Hong, J. (2001). Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research*, *15*, 1–30.

Kautz, H., & Allen, J. (1986). Generalized plan recognition. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 32–37). Philadelphia, PA: Morgan Kaufmann.

Kumaran, V. (2007). *Plan recognition as candidate space search*. Master's thesis, Department of Computer Science, North Carolina State University, Raleigh, NC.

Lau, T., Wolfman, S.A., Domingos, P., & Weld, D.S. (2003). Programming by demonstration using version space algebra. *Machine Learning*, *53*(1-2), 111–156.

Lesh, N., & Etzioni, O. (1996). Scaling up goal recognition. *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning* (pp. 178–189). Cambridge, MA: Morgan Kaufmann.

Nau, D., Au, T.-C., Ilghami, O, Kuter, U, Murdock, J.W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, *20*, 379-404.

Pynadath, D.V., & Wellman, M.P. (2000). Probabilistic state-dependent grammars for plan recognition. P*roceedings of the Conference on Uncertainty in Artificial Intelligence* (pp. 507–514). San Francisco, CA: Morgan Kaufmann.

Sukthankar, G., Goldman, R., Geib, C., Pynadath, D., & Bui, H. (Eds.) (2014). *Plan, Activity, and Intent Recognition*. Cambridge, MA: Elsevier.

Tecuci, D., & Porter, B.W. (2009). Memory based goal schema recognition. In *Proceedings of the 22nd International Florida AI Research Society Conference*. Sanibel Island, FL: AAAI Press.

Vattam, S.S., Aha, D.W., & Floyd, M. (2014). Case-based plan recognition using action sequence graphs. *Proceedings of the Twenty-Second International Conference on Case-Based Reasoning* (pp. 495-510). Cork, Ireland: Springer.

Vattam, S., Aha, D.W., & Floyd, M. (2015). Error tolerant plan recognition: An empirical investigation. In *Proceedings of the Twenty-Eighth Florida Artificial Intelligence Research Society Conference*. Hollywood, FL: AAAI Press.