Bang: A System for Training and Visualization in Multi-agent Team Formation

(Demonstration)

Saulo Antunes Silva[1], Sandro Renato Dias[1], Leandro Soriano Marcolino[2]
[1]Computing Department, CEFET-MG, Belo Horizonte, Brazil
[2]School of Computing and Communications, Lancaster University, Lancaster, UK
sauloantunes@adm.cefetmg.br, sandro@decom.cefetmg.br, l.marcolino@lancaster.ac.uk

ABSTRACT

In this demo participants will explore Bang, a system for multi-agent team formation. Bang automatically selects exercises for training agents, and allows an operator to visualize the expected performance of possible teams, guiding in the agent selection process. Bang is used in the context of programming competitions, a real-world challenge that involves human teams, and significantly improved the performance of the teams of CEFET-MG University.

Keywords

Team formation; Multi-agent training; Visualization

1. INTRODUCTION

Forming strong agent teams is essential in many multi-agent systems domains. It may involve not only selecting a subset of agents, but also training individual agents in order to improve their performance and/or obtain information to aid in agent selection.

We are currently studying the team formation problem in the context of computer programming competitions. Programming competitions are events where teams of students must solve a set of challenging assignments under tight time constraints; the teams are then ranked according to their performance. This is an ideal environment for studying team formation, as it is a real and challenging situation for (human) agents, and agent training and selection are key factors for a successful performance.

Students have limited time available to prepare for programming competitions. Additionally, in general universities have to select a fixed-size set of students in order to comply with competition rules; a decision that must also be taken under time constraints. Therefore, we need a system that selects the best exercises to train the students, and that also enables an operator to quickly estimate the performance of different agent combinations.

Team formation and multi-agent training have received considerable attention recently. Many works focus on when an agent should receive advice from a teacher while being trained [7, 4], but not on which exercises agents should tackle for team formation. Recently, Liemhetcharat and Veloso (2016) [5] proposed an algorithm to select pairs of agents in order to improve their coordination, eventually also forming a team with the best subset. There also has been considerable attention in studying the importance of diversity when selecting agents [6]. These works, however, do not provide a system that automatically selects exercises for agent training, nor visualization tools to aid in agent selection.

We developed a system that allows us to automatically select exercises for agent training, and provides visualization tools to study agent performance and evaluate the expected performance of possible teams, in the context of programming competitions. The system is also able to recommend the team with the best expected performance. By using this system, the programming competition team at CEFET-MG University achieved in 2016 the best result in its history. In this demo, participants of the conference will be able to fully explore our system, including setting up training events and visualizing the performance of potential teams. A video showing our demo is available at `https://youtu.be/ZY0Za8ewA6Y`.

2. OUR SYSTEM

Bang is used to train the individual agents (students), present visualizations to guide an operator in the agent selection process, and suggest the best team. First, we will describe how Bang obtains exercises for training. There are several online systems available containing exercises for computer programming competitions, such as: URI Online Judge [2], UVa Online Judge [3] and SPOJ [1]. Bang works by directly accessing those systems in order to load exercises. The exercises are divided in different categories (e.g., ad-hoc, data structures, graphs, paradigms, strings), are sometimes assigned a difficulty level, and have an associated integer n representing how many agents (in the online repository system) were able to successfully solve it. Bang knows which exercises an agent was able to successfully solve across these different systems, including their associated information (category, difficulty level, number of successful attempts). Agents are also able to see which agents were able to solve a given problem, enabling them to easily seek help for problems that they are unable to solve.

Agents can select which problems they are interested in trying to solve (since we have human agents). However, Bang can also automatically select problems in order to train sets of agents. This happens in the form of *contests*, which is a timed event where a subset of the agents are tested against the same set of problems. As we discuss later, *contests* can be used as an aid in agent selection.

When creating a contest, an operator must manually define how many problems of each category will be part of the contest. This allows an operator to strategize by selecting categories where the agents need training, and/or categories where he/she needs to obtain further information concerning agent performance (in order to select the final team).

Afterwards, agents are able to join the contest. In order to automatically select problems, we are currently using the following

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

algorithm: (1) Form a set of problems **P**, containing all available problems that were not yet solved by any of the agents in the *contest*; (2) Sort P according to the number of successful attempts (in the online repository) n, in descending order, creating a list *L*; (3) Select the required number of problems of each category, following the order of the list *L*. That is, we select the problems that were not yet solved by any agent in the *contest*, and prioritize problems that have a high number of successful attempts.

We use the number of successful attempts as an estimation of difficulty. That is, we assume that harder problems will have a lower n, since less users (of the online database) were able to solve the problem. As mentioned, each problem has an associated difficulty level in the online systems queried by Bang, but we consider n a more reliable estimation of the true difficulty of an exercise, besides not all problems having difficulty level information. Therefore, in the contest we will select problems that are new to all participants, and that are not too hard given their current skills. Additionally, the contest allows an operator to probe which categories the participants are currently strong at.

Bang also shows a graph indicating how many problems have been attempted in the system (in the current week, current month or current year), allowing an operator to estimate the effort of the agents in training.

2.1 Agent Selection

Bang provides visualization tools in order to aid an operator to select agents to form teams for the competitions. These take the form of a radar chart, with one axis for each problem category. When analyzing the performance of an agent, we plot at each axis the number of exercises the agent was able to solve in that category. If the difficulty level of a problem is available, we weight by its value (e.g., one problem of level 2 counts as two problems of level 1). That is, we plot the sum of the levels of all problems solved, and default the level to 1 if the information is not available.

We can also evaluate the estimated performance of a team. Given a subset of agents, we plot at each axis the size of the union of all problems solved by the agents in the respective category (again, weighted by the level). That is, we assume that a team will solve a problem if at least one agent is able to solve a problem, and we remove repeated problems across team members. We can, thus, estimate the best team as the one with the highest area in the radar chart, as the agents would perform well across a variety of problems. In Figure 1 we show an example of the radar graph of a team and their respective individual members. Note that we plot the performance of individual members (Arley, Laura, Mauro) in the same graph, for easy comparison with the expected team performance.

When selecting agents, an operator can use the system to explore the performance of different potential teams. We have a dynamic interface where an operator can click at each potential member, and the expected performance graph automatically updates accordingly. Our system allows, therefore, a human to explore possible combinations, as the operator may have external knowledge that is not yet represented in the system. For instance, friendships and personal compatibility between agents strongly influence the decision process, since we are handling human agents.

However, Bang can also automatically test all possible combinations in order to find the team with the highest radar graph area (albeit computationally expensive as the team size grows), providing guidance to an operator in his/her decision. Additionally, Bang is able to recommend the best team members for a given agent. In order to do so, we find the team with the highest radar graph area when fixing one agent, and considering all possible combinations for the other agents.

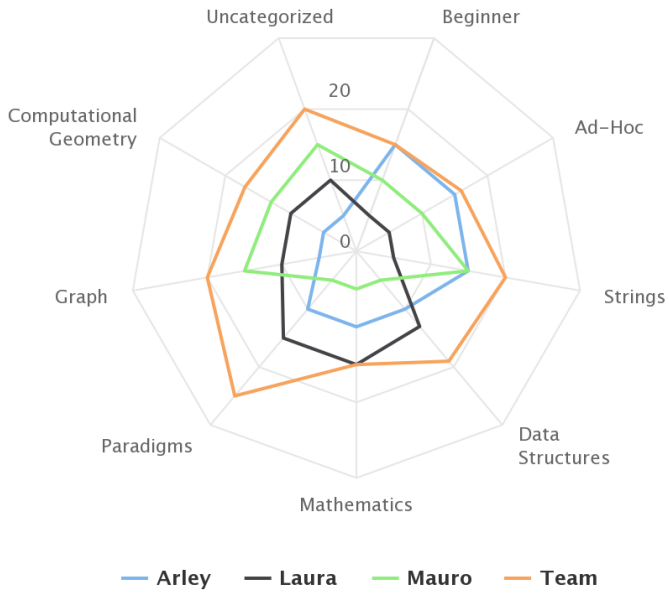

Figure 1: Radar graph showing the expected performance of a team, and of the individual members (Arley, Laura, Mauro).

The system also provides rankings (based on the agents performances while training and in the contests), allowing an operator to quickly identify strong agents, aiding in the decision process. The rankings show the total score of each agent, both in terms of total number of problems solved, and also when weighting each problem by its corresponding level.

Beyond selection, these visualizations also aid in training. An operator is able to quickly assess which categories a potential team or agent is not yet performing well, and use that knowledge when setting up *contests* for training.

The visualization tool can also be used to create teams in the system, and the teams can participate in *contests*. This allows an operator to empirically evaluate potential teams. A ranking is constructed based on the performance of the different teams, helping the operator in his/her final decision.

3. INITIAL RESULTS

We started using Bang to train the programming competitions group of CEFET-MG in 2016. We used the system extensively to train the group members and to select which students would represent the university in actual competitions. The group currently has 40 users, and the operator must select only 3 to participate in the competitions. All members solved around 1000 problems in the Bang system (out of 6500 possible problems in the database).

In the ACM-ICPC South America/Brazil First Phase, the CEFET-MG team was in position 96 out of 598 teams in 2015. After using Bang, we achieved position 67 out of 731 teams, a better position in a much more competitive pool. Thanks to this achievement, CEFET-MG participated in the finals of the ACM-ICPC South America/Brazil competition. In fact, the position we achieved in 2016 is the best one in the whole history of the university.

Acknowledgment: The authors would like to thank CEFET-MG and the School of Computing and Communications for hosting and supporting this research.

REFERENCES

[1] Sphere Online Judge (SPOJ). http://www.spoj.com/. Accessed: 2017-01-14.

[2] URI Online Judge. https://www.urionlinejudge.com.br. Accessed: 2017-01-14.

[3] UVa Online Judge. https://uva.onlinejudge.org/. Accessed: 2017-01-14.

[4] O. Amir, E. Kamar, A. Kolobov, and B. Grosz. Interactive teaching strategies for agent training. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI, New York City, USA, 2016.

[5] S. Liemhetcharat and M. Veloso. Allocating training instances to learning agents for team formation. *Autonomous Agents and Multi-Agent Systems*, pages 1–36, 1 2017. (available online).

[6] L. S. Marcolino, H. Xu, A. X. Jiang, M. Tambe, and E. Bowring. Give a hard problem to a diverse team: Exploring large action spaces. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, AAAI, pages 1485–1491, 2014.

[7] L. Torrey and M. E. Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pages 1053–1060, 2013.