A Genetic Algorithm to Improve Agent-Oriented Natural Language Interpreters

Babak Hodjat[1], Junichi Ito[1], and Makoto Amamiya[2]

[1] Dejima Inc., USA, www.dejima.com
{Babak, junichi.ito}@dejima.com
[2] Kyushu University, Japan
amamiya@is.kyushu-u.ac.jp

Abstract. A genetic algorithm is used to improve the success-rate of an AAOSA-based application. Tests show promising results both in the improvement made in the success-rate of the development and test corpora, and in the nature and number of interpretation rules added to agents.

Keywords: Agent-Oriented Software Engineering, Evolutionary Optimization, GA, Natural Language Interfaces.

1 Introduction

Adaptive Agent Oriented Software Architecture (AAOSA) [1,2] is designed for AOSE Methodology [3,4]. AAOSA is used for Natural Language Interfaces (NLI), where agents represent semantic subdomains of the system (see fig 1). Policies in agents map conditions on input or context of system to claims and actions (see list 1). Policies may need revisions as samples are collected or when the domain of the application is changed or extended.

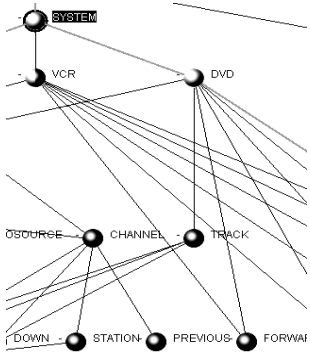

Fig. 1. A cross section of an AAOSA-based NLI for an A/V system.

2 Method, Test, and Conclusions

The GA takes the AAOSA Agent network as the gene to be evolved. Environment is a corpus of sample entries to the NLI system, including desired output actuations.
Mutation adds or removes parts of a policy in an agent. Additions are extracted from failed cases in corpus. Added conditions may be removed through mutation. The existing actions defined in the agent networks are reused in mutated policies.

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 1307–1309, 2004.

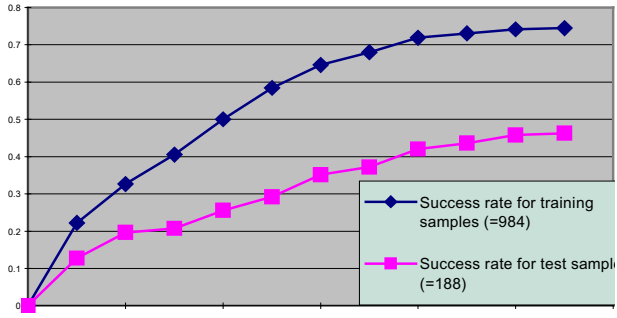

Fig. 2. Plot of success-rate of agent network on training and test data set from 0% to 75% (y-axis) in 5500 generations (x-axis).

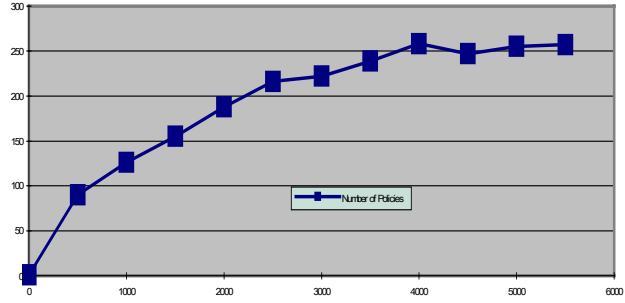

Fig. 3. 257 policies added to agent network after 5500 generations.

Here is a policy from the SOUND agent in the network of figure 1. It has been mutated at least 3 times, once for removing an added policy ('pump'):
*(SOUND: **'ADDED_'/'volume'/'ADDED_' /'del_pump'** | 'sound' {action:{execute '<sound></sound>'}})* Crossover is performed between 2 randomly selected elite pool members. Crossover retains connections and hierarchy of base agent network: *Fitness function = C1 * success-rate – C2 * Number of new conditions*, *C1 > C2 (C1 and C2 are constants)*
To compute success-rate, the samples are run against the gene being evaluated. *success-rate* is number of corpus samples for which the gene produces a correct expected response, divided by total corpus entries. *Number of new conditions* is the difference between number of conditions in the base agent network and the gene being evaluated. This is to minimize the number of new conditions added.
Test results on a production agent network for CRM/FAQ NLI (67 agents 1172 entry corpus, 188 test corpus, 0% initial success-rate) are shown in figures 2 and 3.
Rate and quality of improvements show promise. In tests, GA always improved success rate, close to the best possible success rate achievable given constraints. AAOSA's high encapsulation makes application of GAs possible and practical.

References

1. Hodjat B, Savoie C J, Amamiya M, An Adaptive Agent Oriented Software Architecture, Proc. of the 5th Pacific Rim International Conference on Artificial Intelligence (PRICAI '98) pp.33-46, 1998.
2. Hodjat B, Amamiya M, Introducing the Adaptive Agent Oriented Software Architecture and its Application in Natural Language User Interfaces, In *Agent Oriented Software Engineering*, P. Ciancarino and M Wooldridge eds, Proc. AOSE 2000 conference, Springer-Verlag, pp. 285-306, 2000.
3. N. R. Jennings and M Wooldridge, "Agent-Oriented Software Engineering" in Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press. (to appear), 2000.
4. Iglesias, C.A., Garijo, M., & Gonzalez, J.C. A Survey of Agent-Oriented Methodologies. In Intelligent Agents V—Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98), Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg, 1999.