From: AAAI-99 Proceedings. Copyright © 1999, AAAI (www.aaai.org). All rights reserved.

Using Neural Networks in Agent Teams to Speed Up Solution Discovery for Hard Multi-Criteria Problems

Shaun Gittens
Dept. of Computer Science
AV Williams Bldg.
Univ of Maryland
College Park, MD 20782
sgittens@cs.umd.edu

Richard Goodwin
Jayant Kalagnanam
Sesh Murthy
IBM TJ Watson Research Center
Route 134 and Taconic Hwy
Yorktown Heights, NY 10598
(rgoodwin,jayant,smurthy)us.ibm.com

Hard multi-criteria (MC) problems are computationally intractable problems requiring optimization of more than one criterion. However, the optimization of two or more criteria tends to yield not just one optimal solution, but rather a set of non-dominated solutions. As a result, the evolution of a Pareto-Optimal set of non-dominated solutions from some population of candidate solutions is often the most appropriate course of action. The non-dominated set of a population of solutions is comprised of those solutions whose criteria cannot all be dominated by those of at least one other solution in the current population.

The framework we use, called the Asynchronous Team (A-Team) architecture (Talukdar, Souza, & Murthy 1993), deploys teams of optimizing agents to evolve population(s) of candidate solutions to instances of hard MC problems in order to develop very good solutions. In this framework, agents embody specific heuristics geared to create, modify, or destroy any of a number of possible solutions to a problem instance. These agents are capable of choosing when and on which potential solutions they would like to work on. As a result, as the system progresses in iterations, the population of possible solutions as a whole tends to improve towards a Pareto-Optimal frontier of solutions. The Pareto Optimal frontier would consist of solutions whose individual criteria cannot be further optimized without resulting in a decline in other criteria.

Currently, the method by which each agent chooses to work on particular solutions must be hand coded into the system. It can be very difficult to accomplish this since one would have to determine ahead of time which agents work best on which solutions, requiring much time and effort. In addition, the developer may introduce a 'teacher's bias' to the agent, hand coding incorrect decision-making based on the developer's incorrect analysis of the agents improvement capability. Furthermore, this approach is inflexible as the hand coding done for one problem will not likely be applicable to other problems.

Without hand coding this feature, agents are deployed at random with an equal likelihood. This random deployment, however, often results in CPU cycles being wasted as some agents may be invoked at times when they are unlikely to yield improvements. This problem is significant in that improved decision making by agents should result in the evolution of very good solutions in less time than was previously required.

Thus far we have attempted to remedy this shortcoming by implementing neural network (NN) error back propagation learning techniques (Mehrotra, K. Mohan, & Ranka 1997) to allow individual agents to learn when and on what solutions they work well. We investigate neural network strategies here since they can be trained to approximate smooth functions which output, in constant time, the likely improvement the agent can have on each criterion of a problem instance. One way we implement this is to assign one NN per agent and train it based on the successes and failures that agent achieved over many past runs. The agent's neural network is trained to report a +1 if some preset percentage of improvement (say 10%) in at least one important parameter is expected in the event that the agent is applied to a particular solution in the population. A second, and seemingly more effective, method we use is to train each agent NN until it can estimate the expected improvement the agent could make on a particular solution.

Initial results obtained from testing on tough instances of the Bicriteria Sparse Multiple Knapsack Problem did indeed demonstrate that improved decision making on the part of agents using well-trained neural nets often resulted in significant speedup and overall improved solution quality over the population. More work is being done to overcome drawbacks inherent to neural network solutions such as generalization, input parameter selection, etc. Further study should reveal how solving problems in this fashion fares against the performance of other multiple objective optimization approaches.

References

Mehrotra, K.; K. Mohan, C.; and Ranka, S. 1997. *Elements of Artificial Neural Networks.* MIT Press.

Talukdar, S. N.; Souza, P. d.; and Murthy, S. 1993. Organization for computer-based agents. *Engineering Intelligent Systems.*

Copyright © 1999, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.