Improving the Efficiency of the Distributed Stochastic Algorithm

(Extended Abstract)

Melanie Smith and Roger Mailler
University of Tulsa, Oklahoma
http://www.cnas.utulsa.edu
{melanie,mailler}@utulsa.edu

ABSTRACT

The Distributed Stochastic Algorithm (DSA) is a distributed hill-climbing technique for solving large Distributed Constraint Optimization Problems (DCOPs) such as distributed scheduling, resource allocation, and distributed route planning. The best known version of DSA, DSA-B, works by having agents change their assignments with probability p when making that change will improve their solution (a hill-climbing move). To escape local minima, DSA-B performs a lateral escape move by switching to another equally good value with the same probability p. It is unclear why hill climbing and escape moves are chosen with the same probability. We investigate the performance effects of making these moves with different probabilities, p_H and p_L. Through empirical evaluation, we discover that the efficiency of DSA can not only be considerably improved, but can be more specifically tuned to a particular domain or user's needs when these two move types are considered separately. Our work also shows that DSA can outperform both DBA and DPP when it is properly tuned.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Constrained Optimization; I.2.11 [**Distributed Artificial Intelligence**]: Multi-Agent Systems

General Terms

Algorithms, Experimentation

Keywords

Distributed algorithm, Local Search, DCOP, DSA

1. INTRODUCTION & BACKGROUND

Distributed hill-climbing algorithms are very powerful, practical tools for solving numerous real-world problems including distributed scheduling, resource allocation, and distributed route planning. These problems can be easily mapped to distributed constraint optimization, and like DCOPs, they must be solved using algorithms that can make decisions about how to best improve the global state of the problem from an agent's limited, local perspective. The ultimate goal of distributed constraint optimization is to find a optimal solution while minimizing the overall cost incurred to find it (i.e. the required bandwidth, processing cycles, messaging, etc.)

Complete algorithms are guaranteed to find an optimal solution, but suffer because the cost they incur to find it limits their scalability. So, in practice, one must accept a close-enough solution, especially if the problem is large or the solution needs to be derived quickly. Hill-climbing algorithms tend to work very quickly even on large problems, but do not guarantee that they will find an optimal solution. There are a number of algorithms in this class including the Distributed Breakout Algorithm (DBA) [2], Distributed Probabilistic Protocol (DPP) [1], and one of the most powerful algorithms from this class, the Distributed Stochastic Algorithm (DSA) [4]. One of the greatest benefits of DSA is that it uses considerably fewer messages than protocals like DBA because agents communicate only when they change their values.

DSA is a simple hill-climbing technique that works by having agents change their value with probability p when making that change will improve their solution. The setting of p can have dramatic effects on the behavior of the algorithm and can be quite problem specific. For instance, on very dense problems, having high values of p can cause the protocol to converge more quickly, but the same setting on a sparse problem will cause it to oscillate unnecessarily.

Like other local search techniques, DSA employs a strategy to escape local minima in the search space. One particularly dominant strategy, seen in the DSA-B variant [4], is for an agent to change its value to an equally good value when it detects that it is in a quasi-local-minimum (QLM) [2]. In other words, it will move laterally in the solution space by changing its value, but the overall solution does not get better or worse. In DSA-B, these lateral moves are chosen with the same probability p as hill-climbing moves.

It is unclear why exploration and exploitation moves are chosen with the same probability, so we investigate the effects that lateral and hill-climbing moves have by allowing agents to execute a lateral move with a different probability (p_L) than it would a hill-climbing move (p_H). In this work, we evaluate our DSA variants in two different domains to show that while no single combination of p_L and p_H values perform the best in both domains, explicitly separating these probabilities from one another is a simple change that

Cite as: Improving the Efficiency of the Distributed Stochastic Algorithm (Extended Abstract), Author(s), *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1417-1418

improves the efficiency of the algorithm. For brevity, we only discuss one domain, but results were similar in both.

2. EXPERIMENTATION

To test our DSA variants, we implemented each in a distributed 3-coloring domain. The test cases were compared on two main factors to determine the variant's efficiency - Total Messages Received and Improvement per Message. During each time cycle, the current cost and the number of messages transmitted were measured. These values were used to plot the graphs shown throughout this paper.

Following directly from the definition for a DCOP, a graph coloring problem, also known as a k-colorability problem, consists of the following: a set of n variables $V = \{x_1, \ldots, x_n\}$, a set of g agents $A = \{a_1, \ldots, a_g\}$, a set of possible colors for each of the variables $D = \{D_1, \ldots, D_n\}$ where each D_i has exactly k allowable colors, and a set of cost functions $F = \{f_1, \ldots, f_m\}$ where each $f_i(d_i, d_j)$ is function that returns 1 if $d_i = d_j$ and 0 otherwise. The problem is to find an assignment $S^* = \{d_1, \ldots, d_n | d_i \in D_i\}$ such that $F(S) = \sum_{i=1}^{m} f_i(S)$ is minimized. Like the general DCOP, graph coloring has been shown to be NP-complete for all values of $k > 2$.

2.1 Setup

For this domain, we conducted experiments that have n variables and m binary constraints. The test series consisted of random graphs with $n = \{100, 200, 300, 400, 500\}$ variables and $m = \{2.0n, 2.3n, 2.7n\}$ constraint densities to cover under-constrained, within the phase transition, and over-constrained environments. For each configuration of n and m, 10 graphs were created and 3 different cases (initial colorings) of each graph were run, giving 30 problems that were solved. The random seeds for creating the graph layouts and initial colorings were saved, so each variant solved the same set of problems. Each run was given 500 cycles of execution time. During a cycle, each agent was given the opportunity to process its incoming messages, change its value, and queue up messages for delivery during the next cycle. The actual amount of execution time per cycle varied depending on the cycle, the problem, and the probability values.

2.2 Metrics

Final solution quality is the most obvious metric to compare the efficiency of different algorithms. However, with hill-climbing approaches, the end solution is important, but what sets them apart is their overall efficiency in arriving at an acceptable solution. The solution quality after 500 cycles differs by about 1 for all the algorithms we test, indicating that all produce competitive solutions. In fact, differences are so slight that they are not statistically different, and all were well within a standard deviation of one another.

However, even with such a negligible difference in end solution quality, these algorithms should not be considered equivalently efficient because they converge on their final solution with considerably different costs. Thus, instead of relying solely on solution quality, we use a metric to measure the ratio between cost and benefit by calculating the cost reduction per message. In this way we consider both the quality of the solution being found as well as the overall expense for finding it.

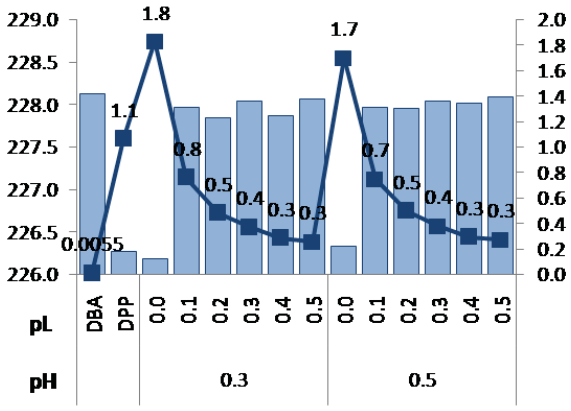

Figure 1: Overall cost reduction (bar) and cost reduction per message (line) for all graph coloring agents for 300 nodes at 2.3n density.

3. RESULTS & CONCLUSIONS

When examining the effects each change in probability has on the solution, we learn that allowing different probabilities to exist allows the user to more finely tune the algorithm to their distinct purpose. From the graph coloring test series, we found that the agent with $p_L = 0.0$ and $p_H = 0.3$ was the most efficient agent (see Figure 1), having a good solution while also using less messaging than DSA with $p = 0.3$. So, for users that want good solution without incurring much expense, avoiding lateral moves is the right answer. If the user is concerned with getting as close to optimal as possible, then using very restricted lateral moves is probably the best option because the solution quality is nearly identical to more aggressive choices while being considerably more efficient and scalable.

When compared to DPP and DBA, the new DSA variants have a 163% and 32727% efficiency improvement respectively. In both cases, the solution it arrives at is within 1% of the solution arrived at by its competitors. It should be clear that although there is a balance necessitated between conflicts and messaging, in some cases, unresolved conflict might be more tolerable than the extra messaging it would take to arrive at a better solution. In the other extreme, messaging may not be a dominant consideration, so higher values of p_L and p_H would be more appropriate.

4. REFERENCES

[1] R. Mailler. Using prior knowledge to improve distributed hill-climbing. In *Proceeding of IAT-06*, 2006.

[2] M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the 2nd Int'l Conf. on Multiagent Systems*, pages 401–408, 1996.

[3] W. Zhang, G. Wang, and L. Wittenburg. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. *Proceedings of AAAI Workshop on Probabilistic Approaches in Search*, 2002.

[4] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Chapter 13: A comparative study of distributed constraint algorithms. *Distributed Sensor Networks: A Multiagent Perspective*, 2003.