EXTENDED ABSTRACT

Approaches to Distributed Optimization : How do we make our agent searches efficient?

Snehit Prabhu
Autonomic Computing
IBM Software Group
snehit.prabhu@gmail.com

We look at the general problem of Distributed Optimization of a system under a set of constraints. We also study the differences between Centralized and Distributed algorithms to the problem. Finally, we present our own novel algorithm to the solve the problem.

The accepted formalism is to model the system on a set of its parameters. Each parameter has a discrete domain in which its values may roam. There exists some "best" configuration values of these parameters at which the system achieves the required goal (notably, optimal behavior). While these parameter scores are available at a single-point for Centralized algorithms, they are not for their Distributed counterparts.

With the centralized paradigm, algorithms may assume that data is cheaply accessible and therefore work on a single node. With larger, heterogeneous and highly interconnected distributed systems, the same assumptions need not hold. Not only is collecting all the data at a single point expensive, it is often times an unacceptable approach. For example, large inter-networks might not be willing to share internal performance characteristics to an outside party for security reasons. Yet monitoring, analyzing and optimizing the function of such huge networks may be beneficial from several viewpoints like better Fault tolerance, Response time, etc.

Centralized algorithms treat the problem as one of Linear programming. Techniques like Convex programming, the Simplex method, etc. fall under this domain. Centrally aggregated data ensures single-point processing. Distributed algorithms, expectedly, work on distributed data. These algorithms work at multiple points (called agents), and may never have the "complete picture" at any one processing point – but rather, have to slowly build such views. Inter-agent behavior is their distinguishing characteristic : agents can be Collaborative or Selfish, process Synchronously or Asynchronously, or any combinations of these. Some distributed algorithms are even hybrids that use these characteristics in different degrees at different stages.

Greedy techniques like Game theoretic algorithms allow the individual agents of the system to optimize selfishly. Information sharing through inter-agent communication addresses the aspect of distributed data. Advantages are usually asynchrony (i.e. fine grained parallelism) and minimal interaction between agents. Disadvantages are that the system may settle at suboptimal solutions, called *Nash Equilibriums*.

Collaborative agent behavior on the other hand sacrifices asynchrony for better search accuracy. Agents aggregating and sharing information in a hierarchical or peer-to-peer arrangement, help the algorithm to iteratively build up better views of the landscape, if not the complete "bigger picture". Once again, subjective issues like trust between agents of the actual system in question must be addressed. Also, extensive collaboration costs must be balanced against those of reaching a suboptimal solution. Another prevalent search family is Clustering – which relies on heuristic measures to narrow the search space. If the scoring landscape is extremely non-linear, these algorithms do not work well.

The score landscapes of distributed systems maybe static, or dynamic (where agent scores are recalculated regularly). Finding optima in the former case is easier than the latter. Our studies show that dynamism is the bane of the distributed optimization problem. Constructing the larger picture by painstakingly sieving

To appear in *Proceedings of IEEE International Conference on Integration of Knowledge Intensive Multi-agent Systems* (KIMAS '07).

through distributed data in a static score landscape is an expensive task. Score fluctuations only aggravate the problem, because the process must be repeated every time scores are reassigned. In this case, the algorithm may run in a loop - perpetually optimizing the overall score as best as it can.

Finally, we present a new hybrid optimization algorithm. The algorithm is part selfish and part collaborative. The system is modeled as a Graph, with the vertices representing the agents. Edges capture the system constraints, while redundant edges maybe introduced to make the graph Connected. We then generate spanning trees of the graph, with vertices belonging to the same layer of the tree performing search asynchronously w.r.t each other. Search results are propagated upwards till the root(s), and search is iterated till a global solution is found. The algorithm is expectedly NP, since the Distributed Constraint Optimization problem is NP-Hard. It offers significant reductions in search space in the average case. Worst case asymptotic behavior degrades to brute force.

We also present a study of when various algorithms maybe applied. Questions like how to sacrifice an individual for the greater good, when democracy beats autocracy (and vice versa), and also how to quantify the "goodness" of an algorithm for the task at hand, are all addressed.

To appear in *Proceedings of IEEE International Conference on Integration of Knowledge Intensive Multi-agent Systems* (KIMAS '07).