17th ICCRTS
Operationalizing C2 Agility

Distributed Algorithms for Resource Allocation Problems

Topic: Networks and Networking

Authors:

Mr. Samuel W. Brett
Dr. Jeffrey P. Ridder

Point of Contact: Mr. Samuel W. Brett

Organization: Evidence Based Research, Inc.

1595 Spring Hill Road, Suite 250
Vienna, VA 22182

Telephone: (703) 287-0371

Email address: brett@ebrinc.com

Title:

Distributed Algorithms for Resource Allocation Problems

Authors: Mr. Samuel W. Brett and Dr. Jeffrey P. Ridder

Paper# XXXXX, Track XXXXX: Networks and Networking

Abstract:
Some of the most challenging problems for decision makers to solve are those that have to do with allocation of resources. These problems are mathematically challenging, and because of the uncertain, uncooperative environments in which they must be solved, there is typically little choice but to resort to manual, ad hoc methods. In this paper we discuss the mathematical nature of these problems and why they are so difficult. We then discuss an emerging family of algorithms based on distributed processing that are particularly well suited for resource allocation in real-world environments. We also discuss our own contribution, an algorithm called Anaconda, which has been applied to bandwidth allocation problems with over 100,000 variables and produced useful solutions within 10 minutes on aging desktop computers.

Introduction

Amongst the most challenging questions for decision makers to answer are these: What and how many resources do I need to accomplish my objectives? How do I ensure that these resources are available when I need them? How do I allocate or schedule my resources? Furthermore, the environments in which these questions must be answered are typically non-cooperative, with incomplete and uncertain information, as well as dynamic, such that the "right" answers are changing with time. Due to the complexity of the questions and the environments, algorithmic solutions are difficult to achieve and, therefore, decision makers commonly revert to manual solutions based on experience. Where algorithmic aids are available, they are typically simplistic in terms of their solution method, choosing instead to emphasize graphical presentations of information to aid sensemaking.

In this paper, we begin by discussing the nature of the mathematical problems underlying each of these questions. The academic literature in mathematics and operations research has long recognized these problems, known as resource allocation problems, to be NP-hard, meaning that these problems are so difficult to solve that there is no existing method to compute an optimal solution in a reasonable amount of time for even modest sized problems. However, we are saved by the fact that in real world applications optimality is rarely needed, and instead we are willing to accept solutions that are better than those that are manually produced. Next, we review some of the traditional solution methods to these problems. Finally, we discuss an emerging class of algorithms using distributed solution methods which show great promise in solving the most challenging of these problems.

Problems

Examples of such problems include scheduling, supply chain management, network design, weapon targeting, sensor networking, and network routing, to name a few. Applications can often be overconstrained so that no global optimum even exists, and even feasible solutions may not be achievable. Whether there are global optimums in any case is not of the greatest concern. Useful descriptions of problems normally have more than one objective. Multiple objectives for a problem cause any solution to trade off in one dimension for one or more other dimensions. Related to this is the concept of a Pareto front as seen in Figure 1, where a solution is only superior to another if it is better in all dimensions.

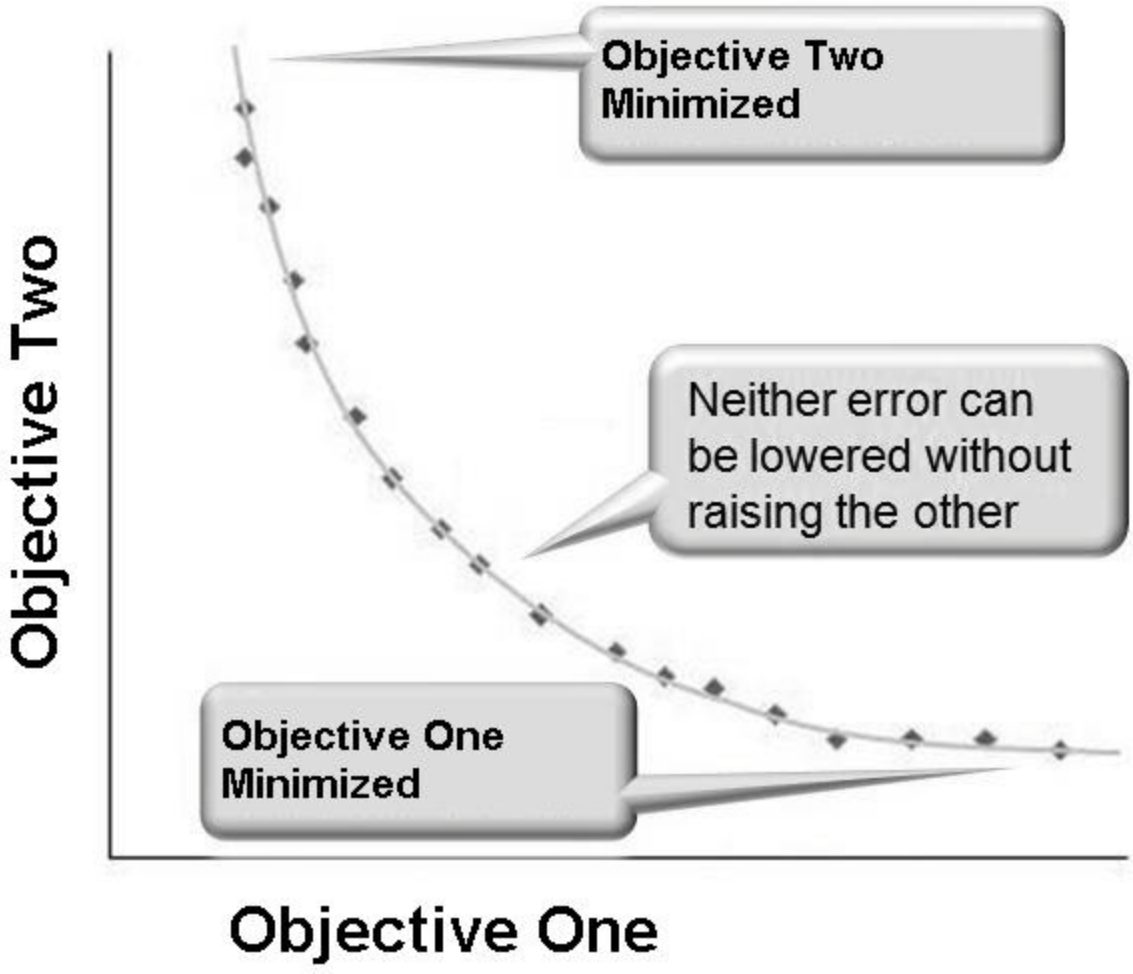

Figure 1: Pareto front with two objectives

The process of scheduling has a wide range of real world applications. In the context of a busy workplace, it can often be difficult to find a suitable work schedule for employees. Consider a hospital where specialists must be present, patients' wait times should be minimized, and uncertainty inevitably arises. Recently, more advanced global optimization heuristics such as genetic algorithms have been applied to this problem. They can often find a good solution to increase coverage of specialists for planned appointments, but adding in the reality of drop-ins needing immediate medical attention quickly throws the system out of balance. What these solutions lack are robustness and flexibility.

A similar example arises in production processes. The well known problem of job shop scheduling considers the optimal way to schedule jobs on a set of machines in a manufacturing plant. There are algorithms to create decent solutions for static assignments. However, the situation of rescheduling where new jobs come into the system while machines are running presents a problem. One choice is to reschedule the system as if it were static every so often. This solution method is undesirable as it can be extremely disruptive to the system and produce

highly sub-optimal results. A dynamic element of the problem to consider is that machines will occasionally fail during production. One way to account for this situation is to build slack into the system, but it is often not clear how to build in this slack. Two sample schedules can be seen in Figure 2.

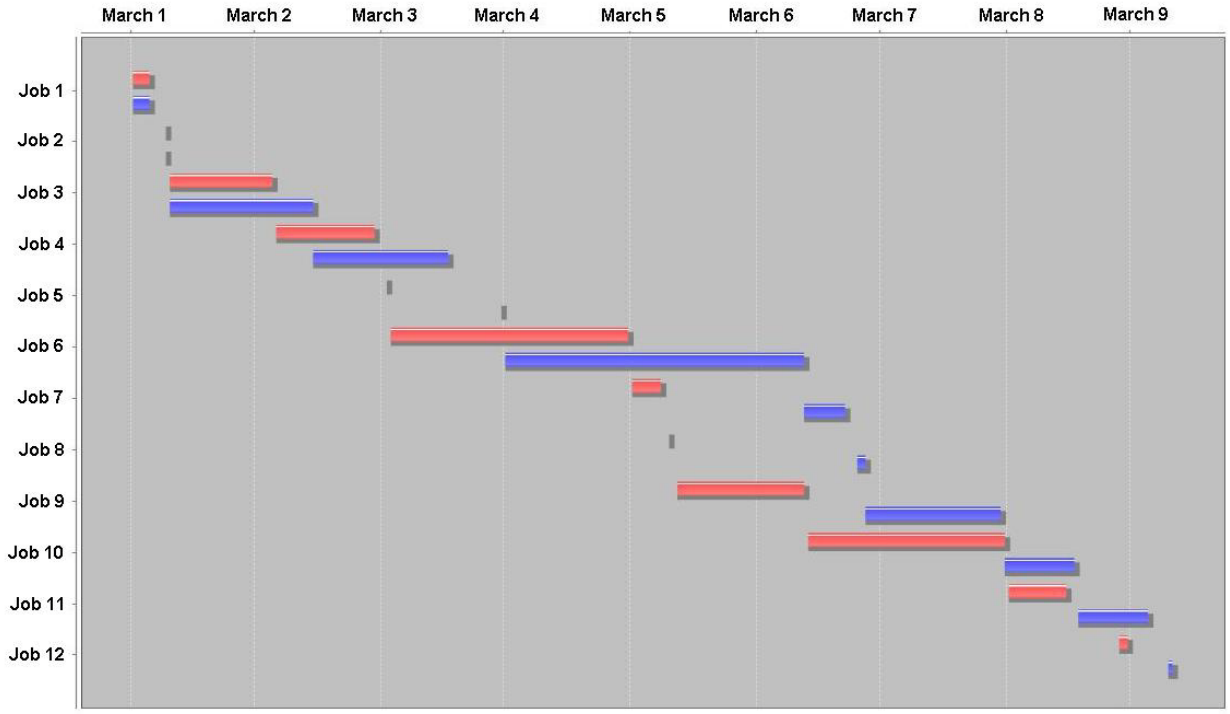

Figure 2: Two sample schedules

The problem of proper supply chain management has long been an extremely challenging problem. Not only do sites have to manage inventory in an uncertain environment but they must also plan how to best route their supply networks. Most inventory planning assumes that lead times from suppliers come from some known distribution. However, a rapid shift in availability of a product could significantly change the optimal strategy. Setting efficient delivery routes is often hard because of the number of possible routes available with differing costs and schedules. Add in any notion of unreliability of a route and these solution methods break down. One way of accounting for this would be to build redundancy into a supply network, but this is costly and where and why is not always clear.

Algorithms

In optimization problems, it is often useful to distribute the problem into smaller sub-problems. This is especially true when your solution space becomes large. There exist non-distributed algorithms that can efficiently compute global solutions to such problems. However, these algorithms will rarely scale well in terms of solution quality, and frequently have difficulty in producing consistently reasonable local solutions even though the global error is small. This is when decomposing a problem becomes useful. The overall solution as well as local areas of the solution are directly accounted for and likely to be reasonable.

This idea has a formal representation known as a distributed constraint optimization problem (DCOP). This problem is quite similar to any other multi-objective optimization problem as it optimizes variables while constraining the feasible region of the full solution. The main difference is that the problem is partitioned into agents. Each agent has a set of variables and constraints to manage as well as a local optimization criterion. The goal of DCOP is to find a feasible solution with the highest ranking by all agents determined by some solution ordering operator.

DCOP has often been applied in the domain of scheduling. One such example is medical appointment scheduling[1]. Here the problem intuitively decomposes into logical entities. For example, they create patient agents to represent patients, their appointments, and constraints and diagnostic unit agents to represent diagnostic units, their workplaces, and constraints. There exist two competing objectives in this domain, optimal resource usage and patient satisfaction. Certainly if every patient had an appointment exactly when they wanted, this would limit the amount of resources utilized. But if appointments were scheduled in a manner that would maximize resource usage for a diagnostic unit, then patients' appointments would each be less convenient.

Another area in which multi-agent distribution of problems becomes useful is in supply chain management. In a simple example, the problem is decomposed into two decision agents[2]. One agent is the production and production inventory planner while the other is the transportation and customer inventory planner. They explore the idea of varying demand and its effects on qualities of solutions for both global and agent based optimization. In the second model, each agent has a built in optimization protocol. They found that when demand estimates were poor, agent based optimization was superior; otherwise, agent based optimization was comparable. So even in smaller examples, distributing optimization problems can yield noticeable benefits.

There are many variations on distributed methods for solving these problems. Algorithms operating synchronously, so that each agent acts in succession, are often less useful than algorithms acting asynchronously due to the benefits of utilizing parallel computing. Some of these methods, referred to as complete, find some solution when one exists and reports nonexistence when present while others, referred to as incomplete, may never terminate[3]. For large problems, finding a global optimum is not computationally feasible. Different schemes exist to prevent getting trapped in local optima such as backtracking, breakout, and varying levels of value commitment. Often the problem can be broken up by use of techniques such as branch and bound. Following are some newer methods along with an algorithm we have developed.

---

[1] Markus Hannebauer and Sebastian M. Distributed constraint optimization for medical appointment scheduling. In *Proceedings of the fifth international conference on Autonomous agents* (AGENTS '01). ACM, New York, NY, USA, 139-140. DOI=10.1145/375735.376026 http://doi.acm.org/10.1145/375735.376026

[2] P. Davidsson, J. A. Persson, and J. Holmgren. On the integration of agent-based and mathematical optimization techniques. In Agents and Multi-Agent Systems: Technologies and Applications, volume 4496 of Lecture Notes in Artificial Intelligence, pages 1–10. Springer, 2007.

[3] Christopher Kiekintveld, Zhengyu Yin, Atul Kumar, and Milind Tambe. 2010. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1* (AAMAS '10), Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 133-140.

The Adopt[4] (Asynchronous Distributed OPTimization) algorithm is unusual in that it allows the user to find the global optimum or to specify what distance is desired from the optimum. This algorithm differs from standard backtracking algorithms in that it can change values when the possibility of a better solution exists, preventing the need for global information to enable backtracking. This construct allows it to be asynchronous.

Another algorithm used for DCOP, although originally applied to distributed constraint satisfaction, is the distributed breakout algorithm[5]. Distributed breakout allows communication only among neighboring agents. This reduces computation and allows asynchronous execution. Neighboring agents will communicate with one another to determine which agent should change its value. This prevents situations where two neighboring agents' values will oscillate. Similar to Adopt, agents do not detect whether they have truly become trapped in a local minimum at any given point. Instead, they can detect if they are caught in what the authors refer to as a quasi-local minimum.

Distributed pseudotree optimization-procedure (DPOP)[6] has been applied in the domains of sensor networks and scheduling with great success. It is an extension of the sum-product algorithm[7] for general DCOP problems. The algorithm first establishes a pseudotree structure for the problem with nodes representing variables. The reason the problem is structured as a pseudotree is that back edges exist. The solving of the problem starts from the leaves of the pseudotree upwards where parents are referred to as target variables. When back edges exist to disrupt this process of obtaining solutions for subtrees, the variables connected by back edges, called context variables, must be factored in to obtain a more consistent solution. DPOP ends up sending a small number of messages, despite some of them being large. They have successfully applied the problem to a large scheduling problem with 200 agents, 101 meetings, 270 variables, 341 constraints.

A recent algorithm based on branch and bound is named no commitment branch and bound (NCBB)[8]. NCBB works by first partitioning the problem so as to enable asynchronous operation. This allows use of parallelism and significantly reduces the cost of agent communication compared to other DCOP algorithms. This algorithm initializes its values using a greedy search. Then it allows each agent to change values with a logical ordering as to prevent the need for backtracking. NCBB has been shown to outperform Adopt in several domains while being dominated by DPOP. However, NCBB's main utility is its low memory footprint.

---

[4] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence, 161(1-2):149–180, 2005.

[5] Yokoo, M., and Hirayama, K. 1996. Distributed Breakout Algorithm for Solving Distributed Constraint Satisfaction. In Proceedings of the Second International Conference on Multiagent Systems. Menlo Park, CA: AAAI Press.

[6] Petcu, A., and Faltings, B. 2005. A scalable method for multiagent constraint optimization. In Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI-05.

[7] Frank R. Kschischang, Bsrendan Frey, and Hans Andrea Loeliger. Factor graphs and the sum-product algorithm. IEEE Transactions On Information Theory, 47(2):498–519, FEBRUARY 2001.

[8] Anton Chechetka and Katia Sycara. 2006. No-commitment branch and bound search for distributed constraint optimization. In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS '06). ACM, New York, NY, USA, 1427-1429. DOI=10.1145/1160633.1160900 http://doi.acm.org/10.1145/1160633.1160900

An algorithm of our own construction named Anaconda (AutoNomous Agent Constraint OptimizatioN Distribution Algorithm) has successfully been applied in bandwidth allocation[9]. One notable difference in Anaconda is its ability to handle continuous as opposed to discrete variables. We represent each variable as an agent trying to locally minimize its errors based on different constraints. Compared to other DCOP algorithms it is computationally inexpensive. It requires memory proportional to the number of variables and constraints and sends messages of minimal size. This fact has allowed us to solve problems with an inordinate number of variables, often on the order of hundreds of thousands, in minutes as opposed to hours or days. Also, with constraints categorized by differing objectives, users can adjust the importance of a given objective in real time. This creates the ability to explore solutions along a Pareto front.

Like other DCOP algorithms, Anaconda is iterative in nature. Variables are initialized to some trivially low value. During iteration, constraints first measure their error based on the values of variables which they constrain. Then each variable adjusts its value by an amount weighted for each class of constraints. In our example, the two classes of constraints are conservation and operational context. The idea is that the constraints indicate how much and in which direction a variable should adjust its value in order to reduce the error on a given type of constraint. The user specifies how much to weight each type of constraint and the limit on how much a given variable can change its variable in a given timestep. In our example, the value of a variable on a later timestep derives from these values:

Here represents the value of a variable (flow value in our example) at a timestep . is the rate by which a variable can adjust itself, named the relaxation rate. is the weight given to a particular class of constraints and is the amount the flow value should change on the given timestep. and represent the two kinds of constraints conservation and operational context, respectively. Despite, or perhaps because ofhaving a simple update equation, the algorithm has been able to provide strong solutions.

Concluding Remarks

Here we have presented a few of the more common resource allocation problems. These problems have direct application to command and control. Most military solutions to resource allocation are done in an ad hoc fashion by human planning. However, finding ways to frame these problems in a more rigorous manner can yield enormous benefits. We have shown how distribution can allow normally intractable problems, due to their large size, to be solved in a reasonable amount of time.

Distribution provides a new method of solving resource allocation problems. Many of these algorithms can be adapted to account for dynamic situations by their agent based nature. The presence of stochastic events, previously limiting the usefulness of classical optimization

[9] Ridder, J., Brett, S., Burris, C., McEver, J., O'Donnell, J., Signori, D., and Schoenborn, H., "Models and algorithms for determining inter-unit network demand," Proceedings of the SPIE Defense, Security, and Sensing Symposium (2012).

techniques, can now be accounted for in order to provide agile solutions. Anaconda allows one to specify tradeoffs among competing dimensions of a problem and view the solution in real time. Applications of these methods in command and control will empower analysts to obtain insights not previously possible.