A Constraint Based Agent for TAC-SCM [⋆] [⋆⋆]

Student: David A. Burke
Supervisor: Kenneth N. Brown

Cork Constraint Computation Centre, Dept. of Comp. Science, UCC, Cork, Ireland
d.burke@4c.ucc.ie, k.brown@cs.ucc.ie

1 Introduction

Supply chain management is concerned with planning and coordinating the activities of organizations across the supply chain, from raw material procurement to finished goods delivery. Constraint programming has been successfully applied to many different aspects of this problem, including production scheduling, inventory management, and vehicle routing. Most of these applications, though, treat the sub-problems as isolated, well defined and static. In practice, all the aspects interact, and the solution process is also complicated by changes to the problem, by uncertainty in the problem description, and by the presence of other agents competing for scarce resources. My research is investigating the role of constraint programming in this wider context. In particular, this paper describes work in progress to build a constraint-based agent for participating in the annual international Trading Agent Competition Supply Chain Management game.

The agent presented combines constraint-based optimisation, reasoning with probabilities, and learning of market conditions in an attempt to determine what customer requests to bid on and what prices to bid. It does this in a manner that maximises its profit while being restricted by capacity and supply constraints. It is implemented in Java, using OPL Studio for the constraint-based optimisation.

2 Background

The TAC SCM [1] [2] game is based around the manufacture and supply of PCs. There are three broad categories of PC and a total of 16 different configurations.

Customers request quotes for making PCs by a given day, agents respond with prices, and then customers select the lowest price offer provided this falls below their reserve price. Customer demand is uncertain. Suppliers provide the components needed to make the PCs. Four types of components are required to build a PC: CPUs, Motherboards, Memory, and Disk drives. Each component type is available in multiple versions and from different suppliers. Agents issue requests for quotes for components, suppliers respond, and then agents make definite orders. Supplier lead times are uncertain.

[⋆] This work is in collaboration with Armagan Tarim, Brahim Hnich and Onur Koyuncu

[⋆⋆] D. Burke is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR)

The agent has a simple production line, where each PC takes a given number of cycles to produce, and there are a maximum number of cycles each day. There are multiple agents in the game, competing for the same set of orders and components. Agents are penalised for running overdrafts, for inventory holdings and for late deliveries. The game lasts for 220 simulated days. At the end of a game, the agent with the highest sum of money is declared the winner.

The TAC SCM game was staged for the first time in 2003. Many different approaches have been tried for competing agents in the two competitions to date. Zhang et al. [3] use game theory and statistical analysis to determine good strategies for their agent, while Benisch et al. [4] use stochastic programming aproaches for bidding and scheduling. Many agents [5] [6] include some method for estimating the probability of offer acceptance in their algorithms.

Constraint Programming has been used in many areas relevant to supply chains. Baptiste et al. [7] showed the effectiveness of constraint-based scheduling. Vehicle routing problems have been solved by combining constraint programming with local search methods [8]. Some existing research also looks at the dynamics of supply chains such as supply chain coordination under changing conditions [9]. Recent research has seen CP used in auctions [10] as well as being extended to handle uncertainty [11] [12] and multi-agent environments [13]. Using a multi-agent approach Calisti et al. [14] have applied distributed constraint satisfaction to both transportation scheduling and resource allocation problems.

3 Implementation

The agent exists in a highly dynamic environment with many factors affecting each decision that has to be made. To simplify the problem, we break the game into three main tasks or decisions that the agent must make at each time step: What offers are made to customers? What orders are made to suppliers? How is production scheduled?–We focus primarily on the first of these tasks.

Here we give brief descriptions of how we deal with production scheduling and supplies procurement, before the main part of the paper which describes our CP model for deciding what offers to make to customers. We then extend the model to dynamically take into account market conditions in the game.

3.1 Scheduling production

A simple approach has been taken to scheduling production. Production is only carried out on confirmed orders. These are ordered by due date and produced if the components exist for them until all the available production capacity is used up. Orders that can not be processed in the current day are provisionally scheduled for being produced the day before they are due to be delivered.

3.2 Making orders to suppliers

The decision on what offers to make to customers is simplified if we already know the cost we paid for components and the numbers of components that are available for use. To do this our agent adopts a policy of ordering components in advance. A number of each component is ordered in advance every day based on expected orders and this amount is adjusted continuously during the game based on the actual orders. We assume all components arrive when scheduled.

3.3 Making offers to customers

Each day, the agent receives a set of request for quotes (RFQ) from customers, specifying *productID*, *quantity*, *reservePrice*, *dueDate* and *penalty*. The agent must decide what customer requests to bid on and what prices to offer for these requests. We want to make these decisions such that profit is maximised.

An OPL model is executed each day to optimise this decision. The decision variable can be modeled in two parts: $bid_r \in \{0, 1\}$ indicates whether or not an offer is made for request r and $price_r$ is the offer price. For now, let us simplify the problem and assume that we will use a fixed price for each product.

In order to determine required supplies and profit, we need to know what components are used in each product, i.e. the bill of materials (BOM). We model the BOM as a binary matrix, with a row for each product i and a column for each component j. A 1 indicates that a component is used in a product, $bom_{i,j} = 1$.

Choosing offers is subject to production ability constraints of the agent. Requests from customers on day t have due dates in the range of $t + 3$ to $t + 12$. In order to deliver an order on a particular date, the order must be completed by the previous day. Furthermore, we will receive orders for offers we make to customers at day $t + 1$. Assuming we only schedule production on receiving the order, then the earliest we can schedule production for these orders is day $t + 2$. Therefore we are looking at our production ability from day $t + 2$ to $t + 11$.

For each day the agent's factory has a certain fixed amount of production capacity available. The capacity required to meet existing orders reduces what is available for each day. The factory capacity, is modeled as an integer array of size 10, one for each day that we can produce products to meet todays requests. The production of computers for a particular day cannot exceed this capacity.

$$\sum_{i=1}^{n} prod_{t,i} \times cycles_i \leq cap_t \tag{1}$$

where n = number of products, $cycles_i$ = number of processing cycles required to build i, cap_t = factory capacity at day t and $prod_{t,i}$ is the number of product i produced on day t, calculated by summing the quantity specified in RFQs that are for product i and that are scheduled[1] for production on day t.

$$prod_{t,i} = \sum quantity_r \times bid_r, \forall r \text{ where } product_r = i, sch_r = t \tag{2}$$

The agents production ability is also subject to component availability. By ordering components in advance, we know the current amount of components

[1] The model provisionally schedules requests for production on a particular day. Actual production scheduling is done once the agent receives confirmed orders.

available, and we also know exactly how much of each component will be arriving at each day. This allows us to add a constraint for availability of supplies.

$$\sum_{k=1}^{t} comp_{k,j} \leq \sum_{k=1}^{t} inv_{k,j} \tag{3}$$

where $inv_{t,j}$ is the inventory[2] for component j at time t and $comp_{t,j}$ is the number of component j used at time t.

$$comp_{t,j} = \sum_{i=1}^{n} prod_{t,i} \times bom_{i,j} \tag{4}$$

The objective function optimises the profit and this requires us to know the cost of producing a product. Using the policy of ordering supplies in advance will allow us to know the current market price for components. Let $price_i$ represent the price of product i and let $cost_j$ represent the cost of component j. This allows us to specify an objective function that maximises our profit where the profit on a product i is calculated by subtracting the cost of components from the selling price, using the BOM to link components to products. In addition to that we can take into account penalties and due dates. It is allowable to deliver a product late but it is subject to penalties. It is possible if the penalty is small enough for us still to make a profit on late deliveries so we could plan to deliver late for a reduced profit. The profit on an RFQ r and the total profit for m RFQs are given in equations 5 and 6 respectively.

$$\begin{aligned} profit_r = quantity_r \times (price_r - \sum_{j=1}^{m} cost_j \times bom_{i,j} \\ -(penalty_r \times max(0, sch_r - duedate_r))) \end{aligned} \tag{5}$$

$$profit = \sum_{r=1}^{m} profit_r \times bid_r \tag{6}$$

We now have a model that roughly captures our problem. But, one major assumption is made. We are using a fixed price and assuming that all offers are accepted. In the game we are competing against other agents, each with individual strategies. The pricing strategy that we adopt should reflect market conditions and in order to manage our production and optimise our profits we need to have an estimate of how likely individual offer prices are to be accepted. We need to be able to reason about the the trade-off between price and quantities. To do this, our agent maintains a probability distribution for how likely an offer with a certain price will be accepted by the customer. If we can successfully predict this probability, then we can allocate our capacity in the best possible way and we can choose a set of offers that will help maximise our profit.

[2] Inventory not used in one day is transferred to the next and added to any newly delivered components.

3.4 Estimating probability of acceptance

How do we choose a price such that it has a certain probability of being accepted? The game specifies *base* prices for each component and the base price of a product can be calculated from this. The customers reserve price falls in the range of 75-125% of the base price of a product. Similarly, the price that we offer will be of a similar magnitude. We define a weight, w, for a product. The price we will offer for that product is the base price multiplied by this weight.

Now, we want to choose the weight such that the offer has our target probability of being accepted. Initially, we can use a default weight and make offers to customers based on this. We keep track of the ratio of offers accepted a to those made o, and periodically update the weight based on this information.

If the actual probability, a/o, is greater than the target then we increase the weight (and hence the price), and if the actual is less than the target we reduce the weight. The update size is relative to the distance between the target and actual value. This means that the further away from the target the actual value is, then the larger the update. This value is multiplied by a step-size factor which decreases over time. This allows us to learn quickly at the start of the game by allowing larger jumps while progressively smaller jumps are allowed as the game progresses until we reach a relatively stable weight value.

Experiments conducted have shown that this results in the weight converging to a point which allows us to choose a price that results in an offer acceptance level that is close to our target probability (See fig. 1). The length of time taken to learn an accurate weight depends on the initial weight and on the step size.

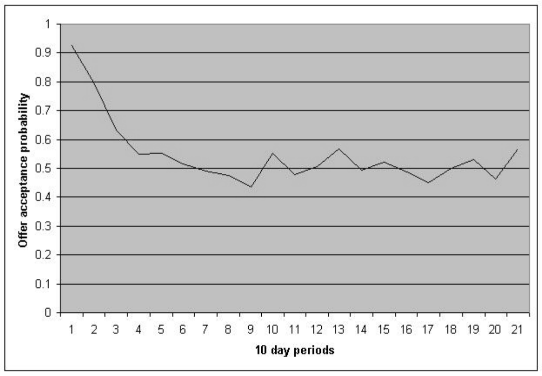

Fig 1. Price learning. The agent learns a price that has an offer acceptance probability close to our target probability (in this case 0.5).

To use this online learning approach in our agent's offer selection problem we learn the price for a range of probabilities and use this in our model calculations. For each product we provide the model with prices that correspond to a configurable number of different probabilities. For each request, the model chooses whether or not to bid, and selects a price from the range. The profit from making an offer as well as the production capacity and components used in fullfilling that offer need to be multiplied by the probability that the offer price will be accepted. The model can then optimise the overall profit, trading off between prices and volume.

4 Conclusions and Future Work

We have presented a supply chain agent for the TAC SCM game that uses constraint programming to decide what offers to make to customers. This agent is currently being implemented and our goal is to enter it in the 2005 TAC SCM competition. Our focus so far has been on choosing offers for customers, and so some simplified decision making is used for other parts of the agent. In particular, further work is needed on an efficient strategy for component procurement.

References

1. SICS: Trading agent competition homepage (2005) http://www.sics.se/tac.
2. Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., Janson, S.: The supply chain management game for the 2005 trading agent competition. Technical Report CMU-ISRI-04-139, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 (2005)
3. Zhang, D., Zhao, K., Liang, C.M., Huq, G.B., Huang, T.H.: Strategic trading agents via market modelling. SIGecom Exchange **4** (2004) 46–55
4. Benisch, M., Greenwald, A., Grypari, I., Lederman, R., Naroditskiy, V., Tschantz, M.C.: Botticelli: A supply chain management agent designed to optimize under uncertainty. SIGecom Exchange **4** (2004) 29–37
5. Ketter, W., Kryzhnyaya, E., Damer, S., McMillen, C., Agovic, A., Collins, J., Gini, M.: Analysis and design of supply-driven strategies in TAC-SCM. In: Trading Agent Design and Analysis Workshop at AAMAS'04, New York (2004) 44–51
6. Pardoe, D., Stone, P.: Tactex-03: A supply chain management agent. SIGecom Exchange **4** (2004) 19–28
7. Baptiste, P., Le Pape, C., Nuijten, W.: Constraint-Based Scheduling. Kluwer Academic Publishers, Norwell, MA, USA (2001)
8. Backer, B.D., Furnon, V., Shaw, P., Kilby, P., Prosser, P.: Solving vehicle routing problems using constraint programming and metaheuristics. Journal of Heuristics **6** (2000) 501–523
9. Beck, J., Fox, M.: Supply chain coordination via mediated constraint relaxation. In: First Canadian Workshop on Distributed Artificial Intelligence, Banff (1994)
10. Holland, A., O'Sullivan, B.: Super solutions for combinatorial auctions. In: CSCLP 2004: Joint Annual Workshop of ERCIM/CoLogNet on Constraint Solving and Constraint Logic Programming. (2004)
11. Fowler, D.W., Brown, K.N.: Branching constraint satisfaction problems and markov decision problems compared. Annals of Operations Research **118** (2003) 85–100
12. Walsh, T.: Stochastic constraint programming. In: 15th European Conference on Artifical Intelligence, IOS Press (2002)
13. Yokoo, M.: Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems. Springer (2001)
14. Calisti, M., Neagu, N.: Constraint satisfaction techniques and software agents. In: Agents and Constraints Workshop at AIIA'04. (2004)