PID Self-Tuning Rules in Agent Design for TAC-SCM Competition

Long Wang
Department of Electrical & Computer Engineering
The University of Texas at Austin

Peter Stone
Department of Computer Science
The University of Texas at Austin

Abstract

The supply chain can be seen as a multi-agent system. Each agent is responsible for one or more activities and coordinates with each other to optimize the performance. TAC-SCM provides a dynamic trading environment which can be used to test different strategies in the agent design. This paper will introduce a strategy which implements PID self-tuning rules and has robust performance in the class-scale TAC-SCM competition so far.

1 Introduction

Supply chain management is very important to a large manufacturing enterprise to meet dynamic market demand changes. Activities such as raw components procuring, production, delivery and order bidding should be planned and executed in a timely and cost-effective way. One new technique is considering a supply chain as a multi-agent system where each agent is responsible for one or more activities and interacts with each other to optimize the system performance.

The e-Supply Chain Management Lab at Carnegie Mellon University and the Swedish Institute of Computer Science (SICS) jointly designed TAC-SCM which provides a testing bed for different agent strategies. Typically, 6 software agents will compete for customer orders and procurement of components from different providers in a 219 days game. Each agent has its own assembly lines and can plan production and delivery everyday. To win the game, agents have to coordinate their behavior in all the activities and may be strategic. Interested readers can find more detailed information about TAC-SCM in [1].

In this agent designed for class-scale TAC-SCM competition, the key strategy implemented is PID self-tuning algorithm which is used in the market for bidding customer orders. The PID control algorithm is widely used for the control of almost all loops in the process industries and is also the basis for many advanced control algorithms and strategies. And the PID tuning technique is used to keep the process staying at the set-point where system optimizes its performance.

In the following sections, section 2 will give an simple introduction of PID control algorithm and PID self-tuning techniques, section 3 will describe the strategy used for bidding customer orders, section 4 will describe the strategy used for procuring components from providers, section 5 will describe the strategy for other activities. And finally, we will evaluate this agent design and expect its performance in the conclusion.

2 PID Control Algorithm

PID (Proportional-Integral-Derivative) control algorithms allow the process control to accurately maintain set-point by adjusting the control outputs. The set-point is where you would like the measurement to be while error is defined as the difference between set-point and measurement. (error) = (set-point) – (measurement) The output of PID controller will change in response to the change of set-point or measurement. Usually there are three control modes:

Proportional Band

With proportional band, the controller output is proportional to the error or a change in measurement.

(controller output) = (error) * 100 / (proportional band)

The proportional band is defined as 100 / gain. There will be a proportional controller offset (deviation from set-point) when using this control mode and increasing the controller gain will make the loop go unstable. Integral control is included to eliminate this offset.

Integral

With integral action, the controller output is proportional to the amount of time the error is present.

(controller output) = (1/INTEGRAL) (Integral of) e(t) d(t)

Integral action can reduce the offset and the load disturbances. However, there is a phase lag between the environment change and the controller output which can be compensated by including derivative control.

Derivative

With derivative action, the controller output is proportional to the rate of change of the measurement or error.

(controller output) = DERIVATIVE (dm / dt)

Where m is the measurement at time t. Derivative action can compensate for a changing measurement to inhibit more rapid changes of the measurement than proportional action. Thus, overshoot can be avoided.

These three control modes can often be used together to stabilize the system fast and accurately.

The most important thing when implementing the PID control algorithm is to understand the underlying process and set correct control parameters. For an unknown system or partly known system, we have to tune these control parameters on the fly which is called self tuning or auto tuning techniques. The typical self-tuning techniques include tuning on demand with upsets and adaptive tuning.

Tuning On Demand with Upset: It will determine the PID parameters by inducing an upset in the process. The controls proportioning is shut off and the control is allowed to oscillate around a set-point. This allows the control to measure the response of the process when environment is changed. From this data the control can calculate and load appropriate PID parameters.

Adaptive Tuning: It will tune the PID parameters without introducing an upsets. When a control is utilizing this function it is constantly monitoring the process variable for any oscillation around the set-point. If there is an oscillation the control adjusts the PID parameters in an attempt to eliminate or minimize them. This type of tuning is ideal for processes where load characteristics change drastically while the process is running.

3 Strategy for Bidding Customer Orders

The strategy for my agent in response to the customer demand is as follows:

Compute the PC prices each day:

Compute the current unit price of each component based on all the offers from the suppliers so far. Then compute the unit price of each kind of PC according to Bill of Material (BOM).

Compute the production status

Sum up the processing cycles in the past 5 days and get the average sumCycles

Compute the profit discount

Use the following function to determine the current profit parameter which will be used in bidding for orders:

Profit +=alpha* (sumCycles–cycleAmount) / cycleAmount

Where alpha has the initial value of 0.08 and will be self-tuned according to the oscillate behavior. That is

Alpha = alpha*cycleAmount/|sumCycles – CycleAmount|

I first set cycleAmount to 2000 which is the production cap for each agent and finally change it to 1800 to avoid high penalty.

Bidding the order using profit parameter

When bidding for the order, my agent will submit the bid as follows:

(bidding price) = (current price) * profit

Protection Mechanism

Most of the time, if the profit goes below 1.0 I will set it to 1.0 which means I am reluctant to sell cheaper than the cost if there was price war. However, in the end of the competition (after day 190), this limitation is removed. (In fact, this mechanism has been changed just before the game. That is, during the middle of the game, the profit parameter is allowed to be as low as 0.5)

Whenever the inventory of some components is below 500 in consecutive 2 days, its price will be set to a high value (5000) to avoid bidding the PC composed of such components. After new ordered components from the suppliers arrive, the alarm will be reset. So, in the beginning of the game, the agent will not bid order from the customers until we have enough required components in the inventory.

4 Strategy for Procuring Components

The strategy for this part is changed often during the agent design due to the fact that we have no inventory cost and every agent my get benefit by placing large request in the first game day. Since there is no reason to doubt that some agent will not do so, the direct result is raising the price of components in the following days to a much higher level. So, my agent will not place order in this period (from day 1 to day 40). Because the capacity of suppliers in the first day is randomly chosen, my agent will supplement the inventory in day 70. Because the demand for each kind of PC change a lot in a game and between games, my agent will supplement the inventory once again in day 130.

First day procurement

Order 10000 units of cpu and 20000 units of other components which are supposed to arrive after 40 game days. For components which are supplied by more than 1 suppliers, each supplier will be requested 10000 units.

Procurement in day 70

The request quantity in game day 70 will be computed using following function

(quantity) = 7000 – inventory (for cpu)

(quantity) = 14000 – inventory (for other components)

Still, when components have two suppliers, half amount will be requested to each.

Procurement in day 130

Just like day 70, replace the 7000 and 14000 with 4000 and 8000 instead.

Regular procurement

Compare the difference needed components with the sum of ordered and inventory, order 200 components in 10 days if the result is positive. When such components have two suppliers, the cheaper one will be selected. To achieve high reputation, I don't send the same requests to both suppliers. Selection is determined based on the price of the last order. So my agent will order 1 component everyday to each agent to keep track of the updated price.

Protection mechanism

Whenever the inventory goes below 500 (the alarm is set at the same time in the customer market), my agent will order 50 units of components in case some supplier suspend delivery due to capacity reduced. The reason is, whenever the delay happens, the order is given priority over orders with a later due date.

To increase the chance of requests being selected by suppliers, one large request is divided several small requests.

5 Strategy for Other Activities

The other activities include production and delivery. It turns out that in current competition environment they are not much important compared to bidding for orders and procurement of components. So I just use the base strategy of base agent here. Actually, these parts are the future work that my agents need to be improved.

Production

All the orders are sorted according to the profitability and puts late order first. The production is executed according to this order.

Delivery

Good delivery plan can be used to minimize the penalty. However, I have nearly no time to implement good delivery strategy, so just deliver the products whenever they are produced.

Bank & Market report

For the same reason, this agent does not consider their effect to the market and agents and leaves them there.

6 Conclusion

Our goal is to maximize the overall gain in utilities. Since the behavior of other agents is unknown and there is some randomness in the virtual economical environment, the strategy must be easier to be tuned to adapt to different games and opponents. PID control algorithm is one of such methods that can achieve robust performance in a timely and cost-effective way.

To analyze the strategy for my agent, let us considering the following scenarios,

Bad market

In the worst case, I will accumulate 10000 units of CPUs and 20000 units of other components in the inventory in the end. This may not happen because every agent is rational enough and will not submit an unreasonable bid. Besides, I have a protection mechanism that will deplete the inventory after day 190 as many as possible.

Another situation might be some components are suspended for delivery and no PC can be produced. In this way, I can not win order but will not get penalty due to the alarm protection mechanism.

So, most of the time, my agent will get a positive profit or a little below zero due to the negative bank interests.

Good market

Market is good means every agent may get enough orders from the customers and get enough components from the suppliers with a reasonable price. In this case, the capability of production of my agent will be fully utilized and will beat down those agents who are also myopic. Here myopic agents are those agents who don't or can't predict future market changes and just bid based on past information.

Those agents who can accurately predict market trend can perform better, however, with the cost of some risk. And it turns out that accurately prediction is very hard to achieve in a relative short development period.

Complex market

Here complex market means customer demand and capacity of supplier change a lot during one game or between games. My agent will go above zero because it can idle in the bad time and be active when there is profit space. Supplemental procurement in day 70 and day 130 allow the agent have enough inventory with low risk of stock large amount of unused components. Penalty tends to be low because most of the time I will have enough inventory and avoid competition in the bad time (from day 1 to 40 and the end of the game). The most important thing is that the PID control parameters are self tuned and can stabilize the system as soon as possible.

In summary, after implementing PID self tuning control algorithm in the strategy, this agent is expected to have robust performance in the coming class-scale competition.

References

[1] Arunachalam, Eriksson, Finne, The TAC Supply Chain Management Game, 2003