Empirically Testing Decision Making in TAC SCM

Erik P. Zawadzki

July 23, 2007

Joint work with Kevin Leyton-Brown

Outline

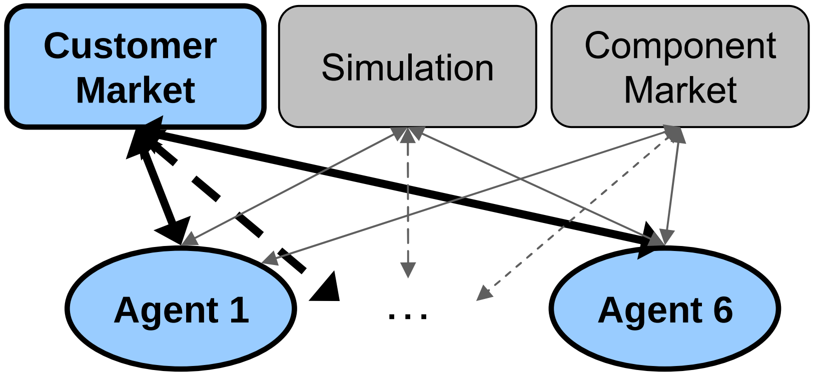
- Introduction to Problem
- Model
  - Customer Market Process
  - Component Market Process
- Application: Scheduling
  - Agents
  - Experiments
- Conclusions

Trading Agent Competition Supply Chain Management (TAC SCM)

- Supply Chain Management (SCM) is an important industrial issue
- Static and unresponsive SC policies
  - Large inventories
  - Unreliable deliveries
  - Underperformance
- TAC SCM
  - Encourages research into SCM solutions
  - A simpler setting

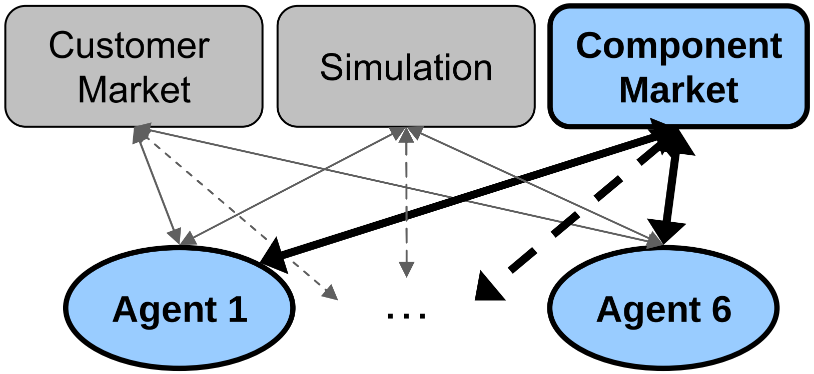
Subproblems

- A TAC SCM PC (personal computer) manufacturing agent must make decisions for the following four subproblems:
  - **Customer Bidding**



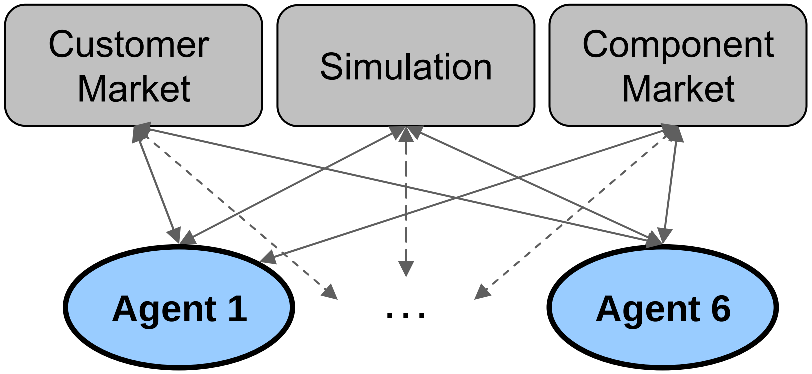
Subproblems

- A TAC SCM PC (personal computer) manufacturing agent must make decisions for the following four subproblems:
  - Customer Bidding
  - **Component Ordering**



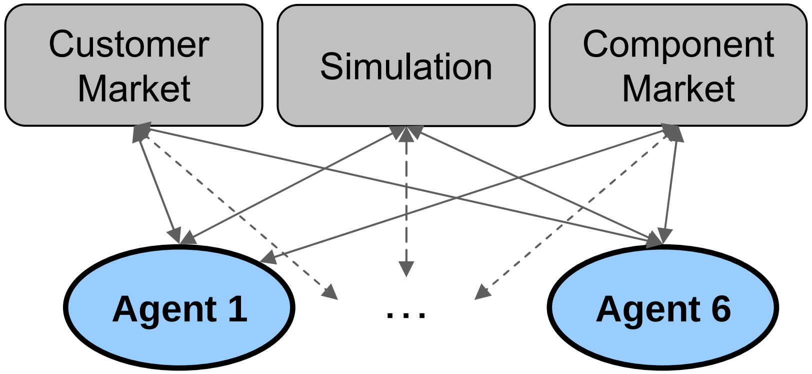
Subproblems

- A TAC SCM PC (personal computer) manufacturing agent must make decisions for the following four subproblems:
  - Customer Bidding
  - Component Ordering
  - **Production Scheduling**



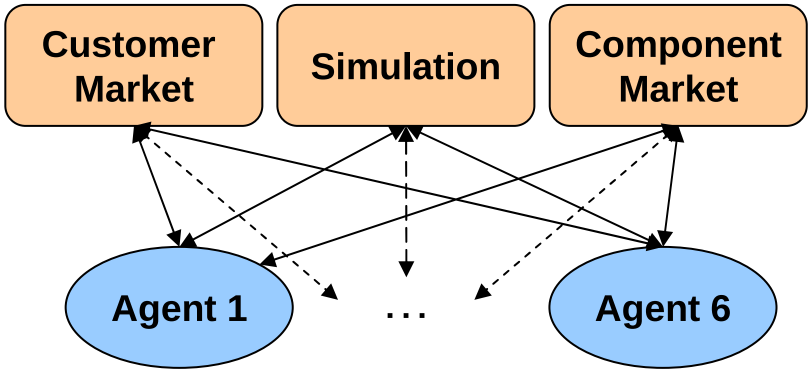
Subproblems

- A TAC SCM PC (personal computer) manufacturing agent must make decisions for the following four subproblems:
  - Customer Bidding
  - Component Ordering
  - Production Scheduling
  - **Delivery Scheduling**



Subproblems

- A TAC SCM PC (personal computer) manufacturing agent must make decisions for the following four subproblems:
  - Customer Bidding
  - Component Ordering
  - Production Scheduling
  - Delivery Scheduling
- Decomposition
  - For instance [Collins *et al* 2007]



Decision Making is Hard

- Decision making in TAC SCM is hard
  - Each subproblem solution influences the other three
    - E.g. Customer Bidding
      - Depends on Delivery Scheduling
      - Depends on Production Scheduling
        - Depends on Component Ordering
  - There is an uncertain future
    - Customer RFQs
    - Component availability and pricing
    - Late component deliveries
  - There is a hard time-constraint
- Most agents simplify or approximate this decision (or both).

Many ways to simplify

- Subproblem connection
  - Introduce independencies
- Action
  - Only build PCs once an order is certain
- Information
  - Do not use all the information that can be collected

How Should Algorithms be Compared?

- How do we determine which approaches are better than others?
  - The traditional approach is running an agent against a large set of other agents
    - Easy to compare complete agents
      - Harder to compare particular approaches to the subproblems
    - Test results are immediately relatable to competition performance
      - Results may be highly variable
        - Multiagent
        - Randomness in the simulation

An Alternate Approach to Evaluation

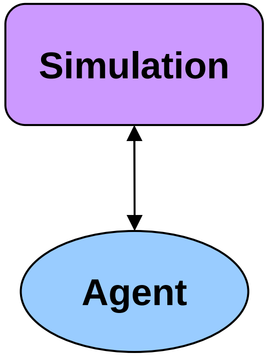
- We suggest a testing framework makes it easy to:
  1. Hold some subproblem algorithms fixed while varying others
  2. Large number of experiments
     - Parallelism
  3. Control variance
     - Blocked experimental design
  4. Focus on particular game events
     - Resource shortages
     - Steady state
     - End game

Outline

- Introduction to Problem
- **Model**
  - Customer Market Process
  - Component Market Process
- Application: Scheduling
  - Agents
  - Experiments
- Conclusions

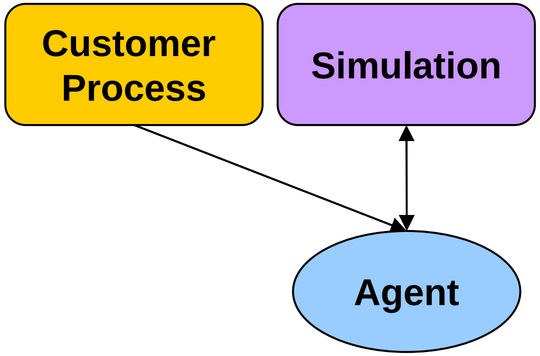
Model Overview

- Our Model:
  - **Generate RFQs and handle factories like in TAC SCM**



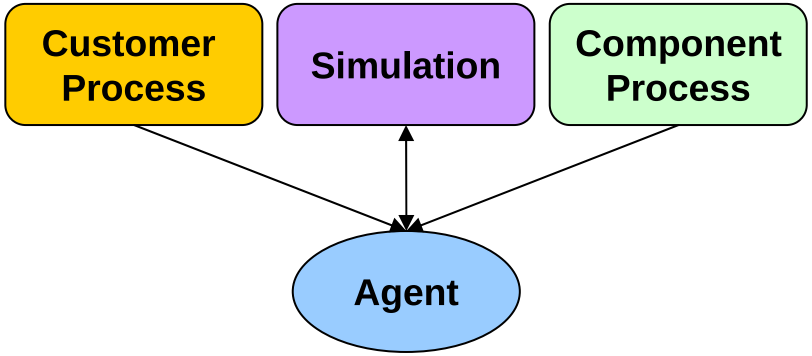
Model Overview

- Our Model:
  - Generate RFQs and handle factories like in TAC SCM
  - **Simulate the customer market using a process learned from game data**



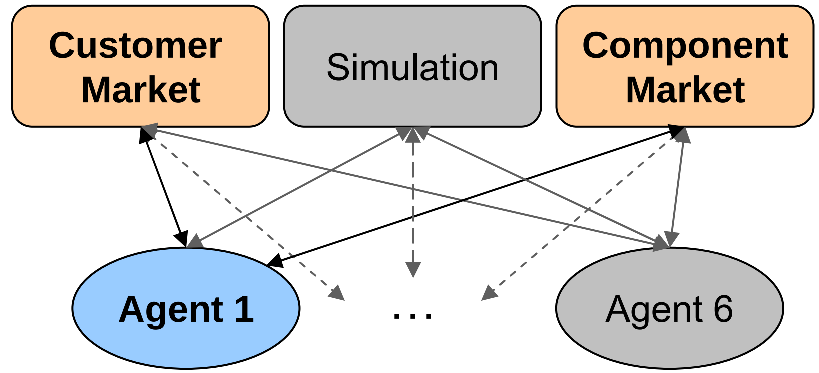
Model Overview

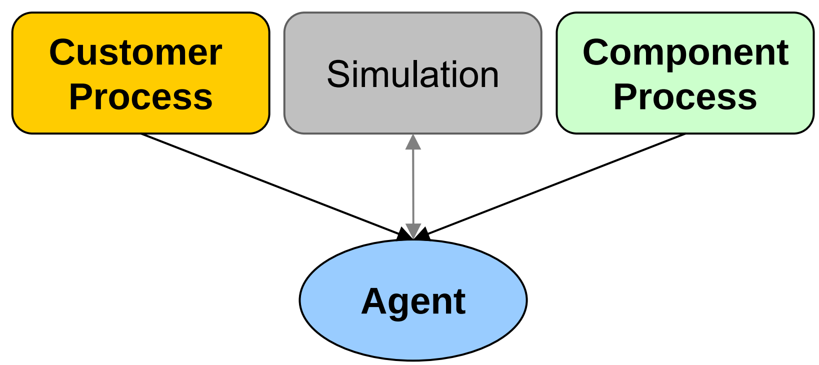
- Our Model:
  - Generate RFQs and handle factories like in TAC SCM
  - Simulate the customer market using a process learned from game data
  - **Simulate the component market using a process structurally similar to TAC SCM's market**



Model Overview

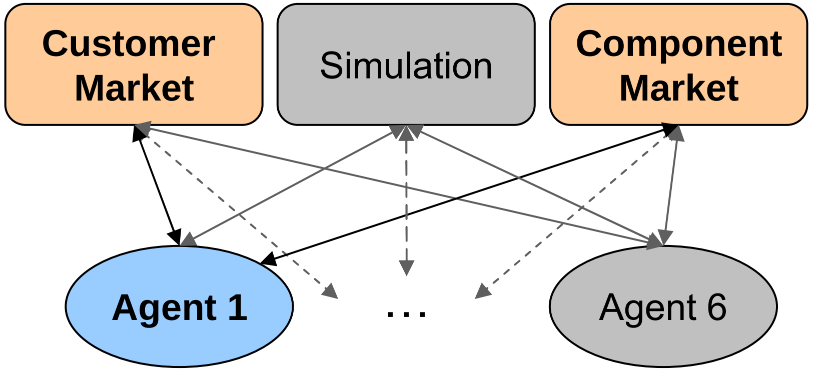
- Processes independent of agent actions
  - Blocked experimental design
    - Simulation defined random seed
    - Block experiments by simulation seed



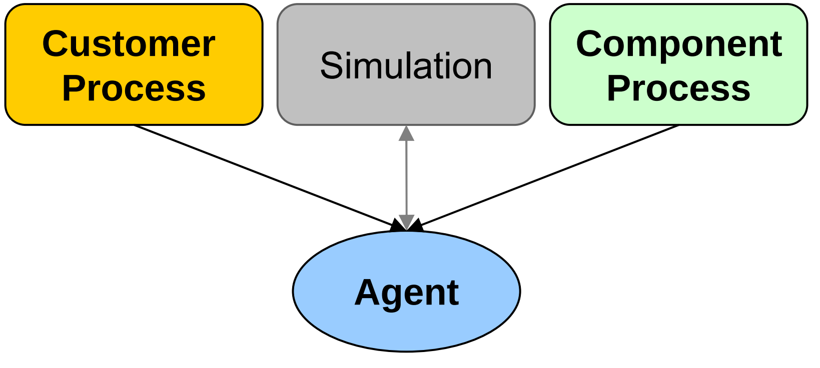


Model Overview

- Processes independent of agent actions
  - Blocked experimental design
    - Simulation defined random seed
    - Block experiments by simulation seed



- We will focus on 'steady-state' behaviour
  - Days 40 to 200
  - Beginning and end game effects
- We need to validate our model
  - Want processes to be faithful to game log data



Outline

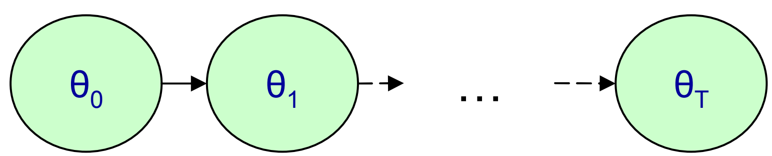
- Introduction to Problem
- **Model**
  - **Customer Market Process**
  - Component Market Process
- Application: Scheduling
  - Agents
  - Experiments
- Conclusions

Customer Market Process (CMP)

- Learn the winning price distribution $p(B|\theta_t,S)$
  - θ_t is the model parameters for day t
  - S is the product type r.v.
- Assume that each day's winning price distribution is a Gaussian

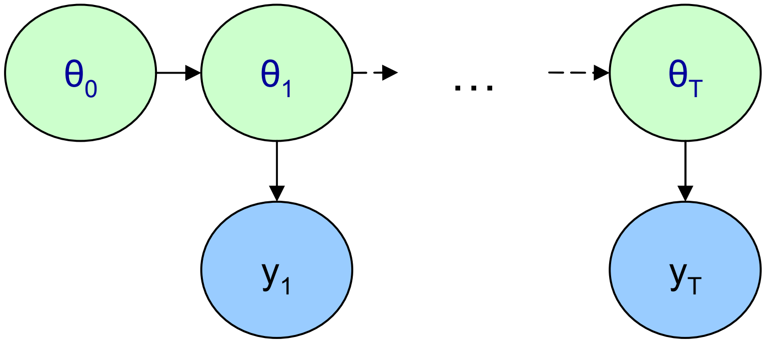
Customer Market Process (CMP)

- Model parameters linearly related to previous day's with unbiased Gaussian noise
  - $\theta_t = A\theta_{t-1} + N(0,Q)$



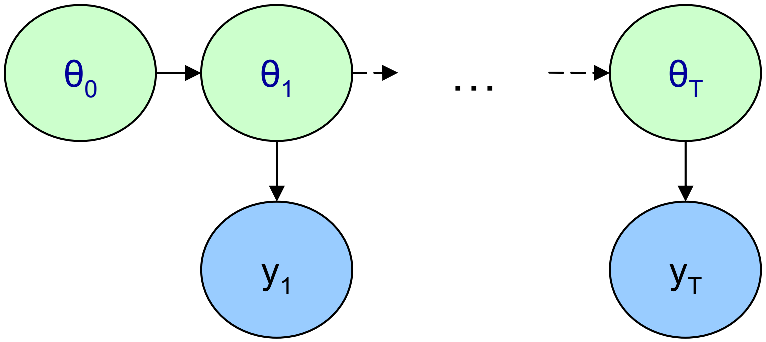
Customer Market Process (CMP)

- Model parameters linearly related to previous day's with unbiased Gaussian noise
  - $\theta_t = A\theta_{t-1} + N(0,Q)$
- Observations (empirical distribution) linearly related to model parameters with unbiased Gaussian noise
  - $y_t = C\theta_t + N(0,R)$

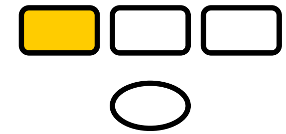


Customer Market Process (CMP)

- Model parameters linearly related to previous day's with unbiased Gaussian noise
  - $\theta_t = A\theta_{t-1} + N(0,Q)$
- Observations (empirical distribution) linearly related to model parameters with unbiased Gaussian noise
  - $y_t = C\theta_t + N(0,R)$
- Linear Dynamic System (LDS)

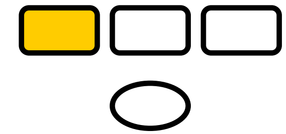


How to Learn LDS Parameters



- Learn the LDS dynamic ($<A,C,Q,R,\theta_0>$) with EM
  - Iteratively improves on an initial model
    - 'Improvement' is increasing data likelihood
  - Unstable
    - Inversion
    - Good initial model helps avoid problems
  - Can calculate data likelihood and predict future states using Kalman Filters (KFs)
    - Recursive filter for estimating LDS states
    - Very fast
    - Simple to implement

Other LDS consideration

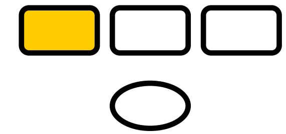


- Other decisions about the model:
  - Independent vs `Full’ Model
    - Should the behaviour from other PCs be informative?
    - Can an LDS model this relationship?
      - Overfitting
  - Different dimensionality of the model parameters

Picking an LDS Model

- What makes a good model?
  - Model easily explains historical data
    - Data likelihood
  - Predictive power
    - Absolute prediction error

Picking an LDS Model

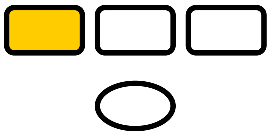


- What makes a good model?
  - Model easily explains historical data
    - Data likelihood
  - Predictive power
    - Absolute prediction error

Model Log-Likelihoods

	Independent	Full
1/16	-35000	-50000
2/32	**-34000**	-94000
3/48	-34000	-144000

Picking an LDS Model

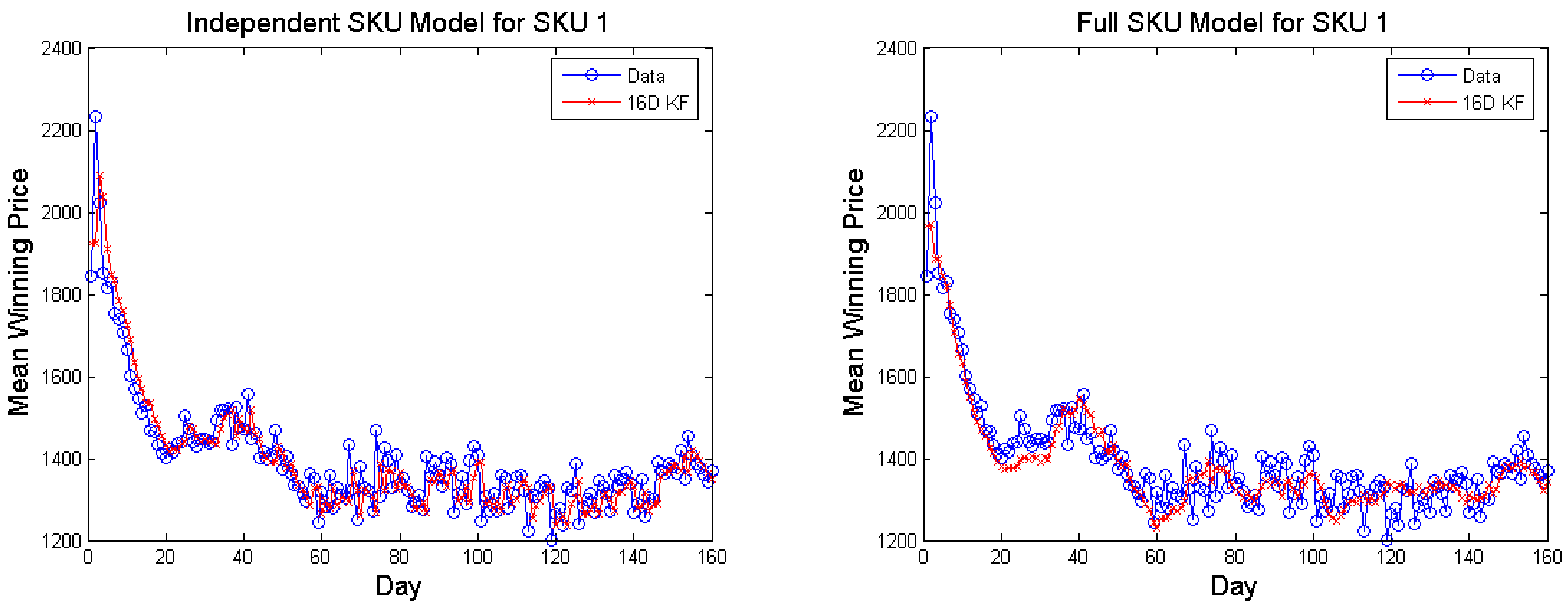


- What makes a good model?
  - Model easily explains historical data
    - Data likelihood
  - Predictive power
    - Absolute prediction error
- Independent Model with 32 model variables
  - Highest likelihood
  - Low mean winning price prediction error of 57.0 units

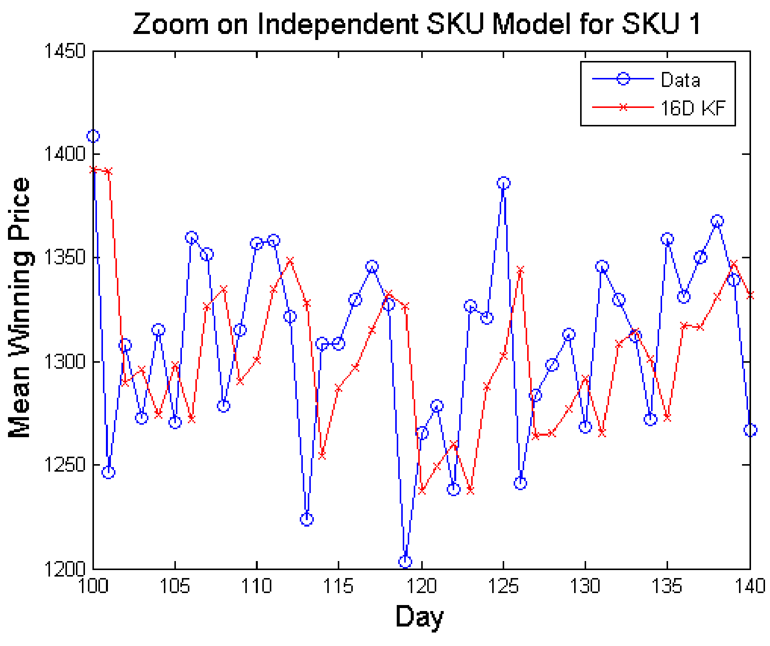
Model Log-Likelihoods

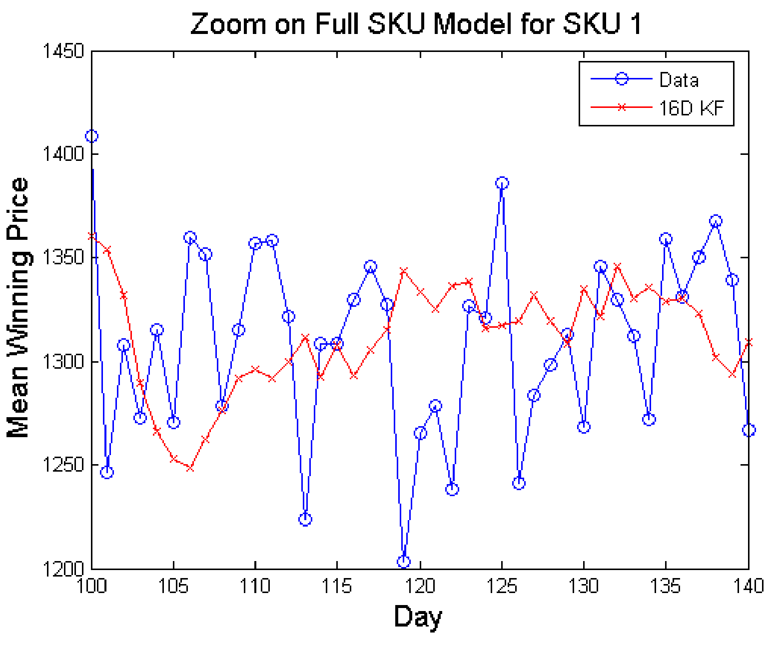
	Independent	**Full**
1/16	-35000	-50000
2/32	**-34000**	-94000
3/48	-34000	-144000

KF Predictions Based on Learned LDS



KF Predictions Based on Learned LDS





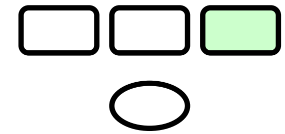
Alternate Approaches

- Our model is generative
  - But this is similar to the prediction problem
- Deep Maize Forecasting [Kiekintveld *et al*, 2007]
  - K^{th}-nearest neighbour
    - What in the past looks like what is being seen right now?
- TacTex *Offer Acceptance Predictor* [Pardoe and Stone, 2006]
  - Separated Particle Filters

Outline

- Introduction to Problem
- **Model**
  - Customer Market Process
  - **Component Market Process**
- Application: Scheduling
  - Agents
  - Experiments
- Conclusions

Component Market Process



- Not the focus of our work
- Needed a simple model
- Made one based on structural similarity to TAC SCM
  - Daily manufacturing capacity determined by random walk
  - Each component manufacturer maximized daily outgoing components using an ILP

Outline

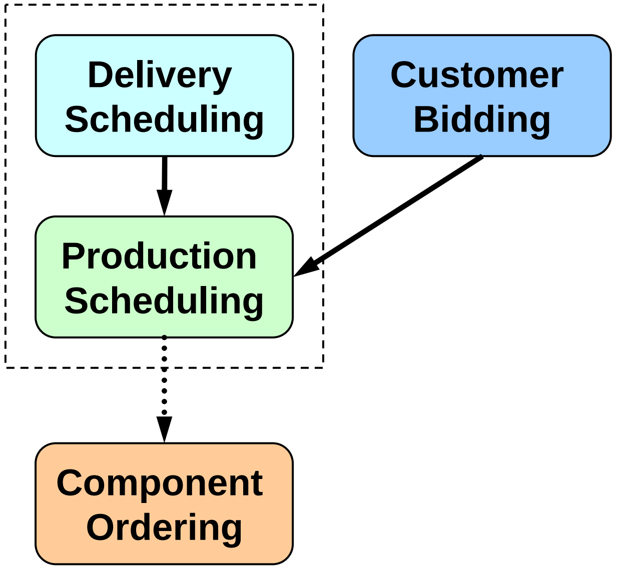
- Introduction to Problem
- Model
  - Customer Market Process
  - Component Market Process
- **Application: Scheduling**
  - **Agents**
  - Experiments
- Conclusions

Comparing Three Scheduling Techniques

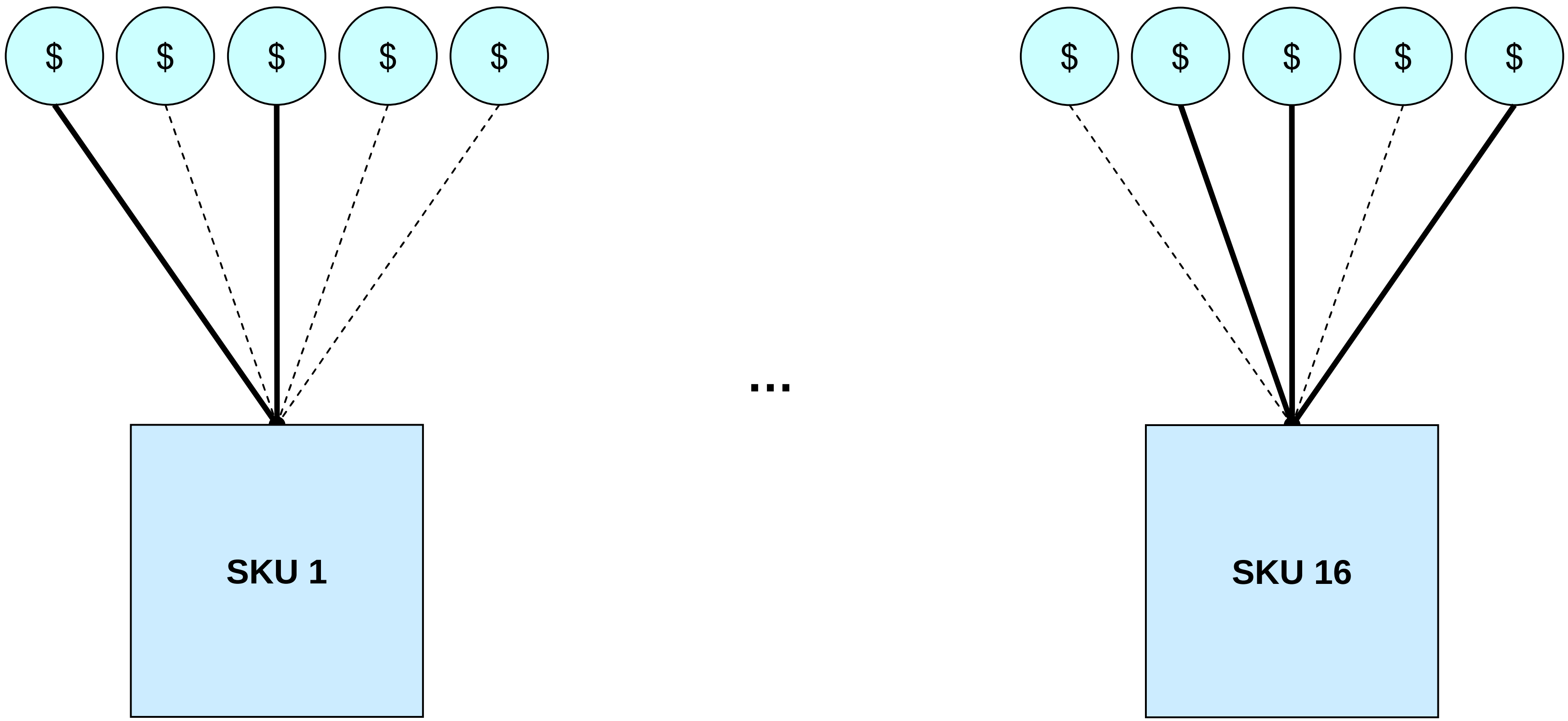
- We will use our test framework to compare three different scheduling algorithms
  - We are interested in the interaction between production and delivery scheduling
  - To maintain consistency, will using the same customer bidding and component ordering algorithms
    - Both done with simple heuristics
      - Ordering: static daily amount with inventory cap
      - Bidding: greedily, fixed percentage of production capacity

Myopic

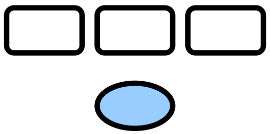
- **Myopic**
  - Delivery Scheduling
    - ILP that maximizes current day's revenue
    - Ignores the future
  - Production Scheduling
    - Greedy, based on outstanding PC demand



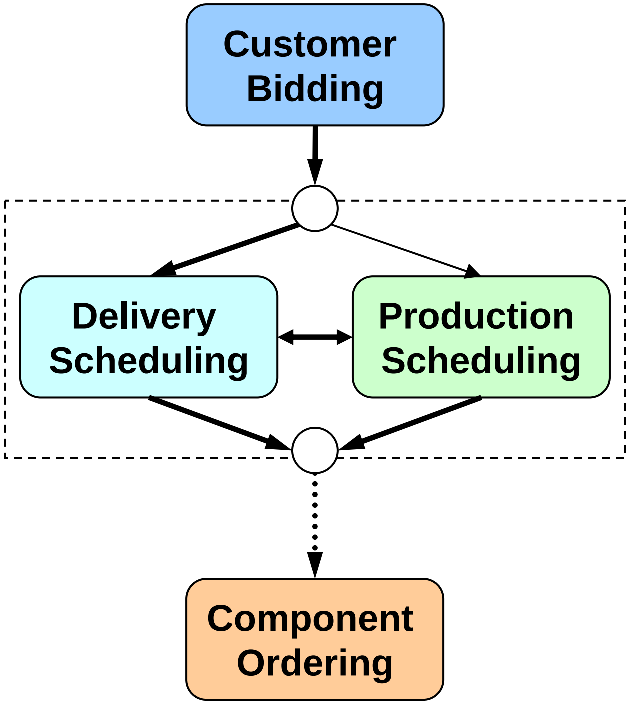
Myopic Delivery Program



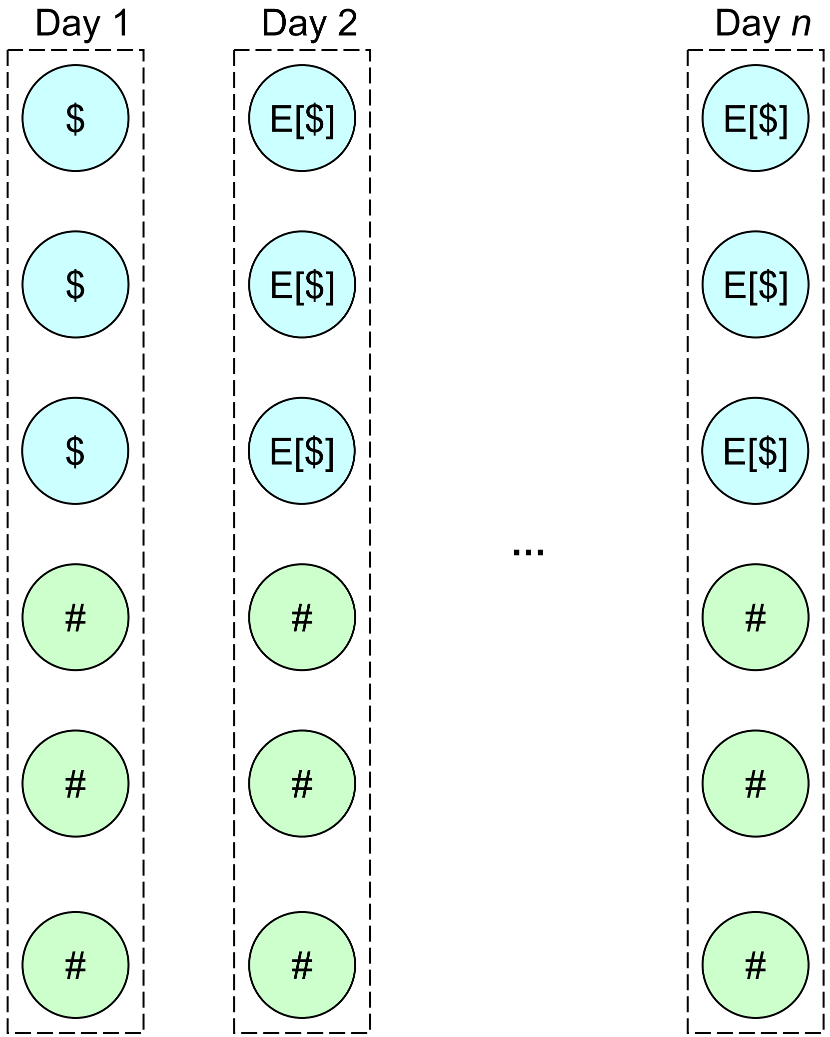
SILP



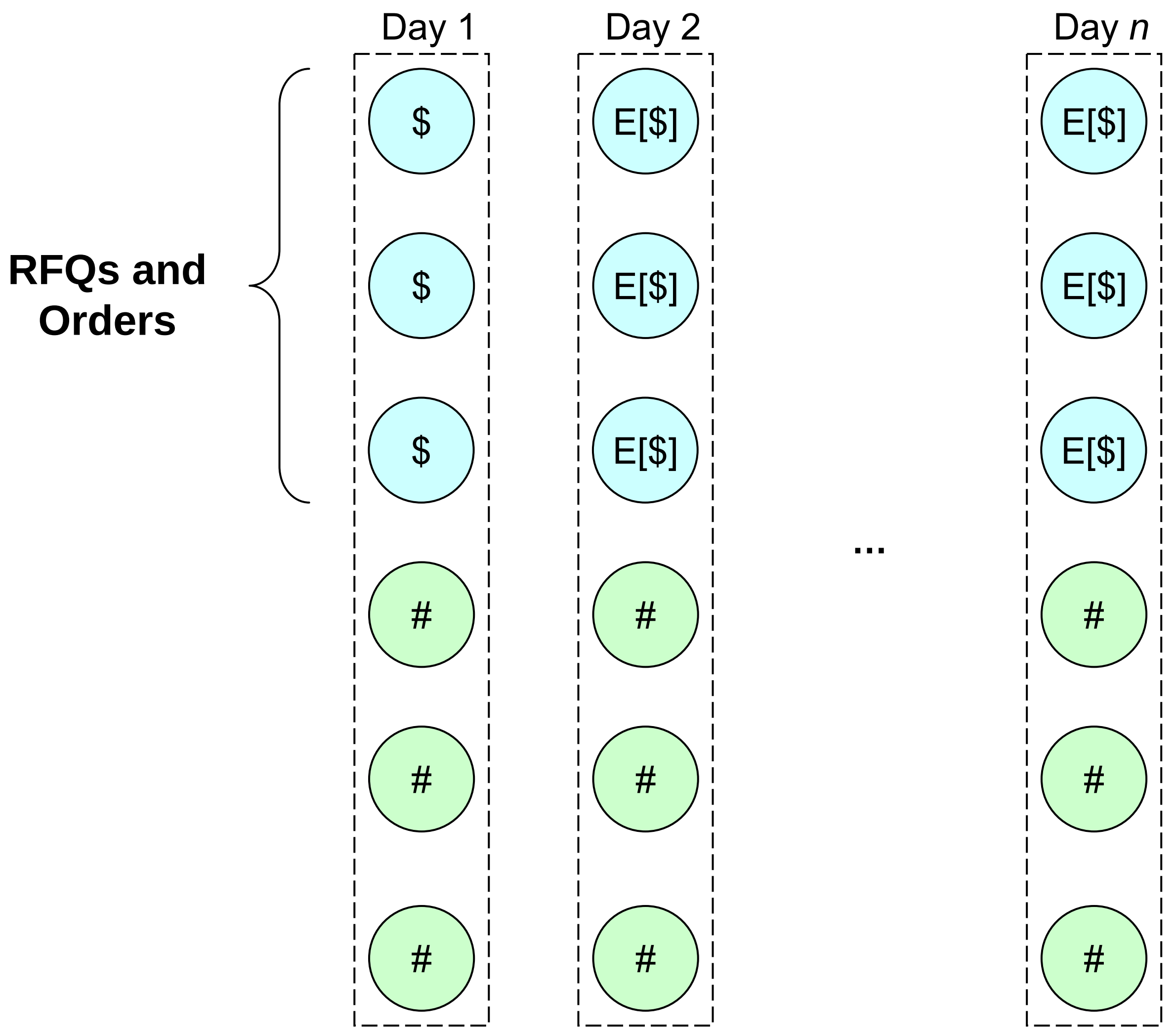
- **Stochastic Integer Linear Program (SILP)**
  - SILP from Benisch *et al* 2004
  - Delivery and Production Scheduling
    - ILP that maximizes expected profit
    - Fixed *n*-day horizon



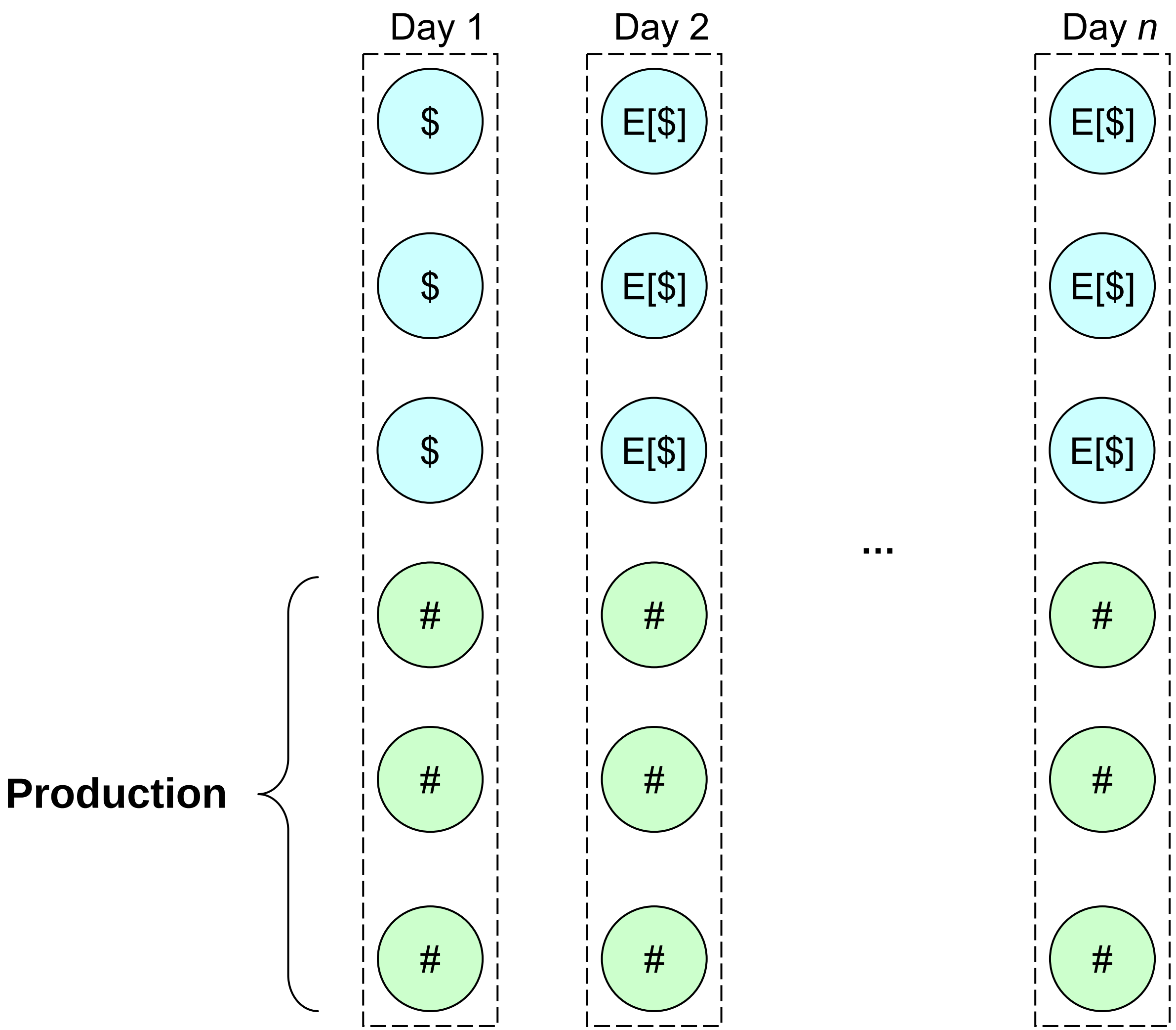
SILP Program



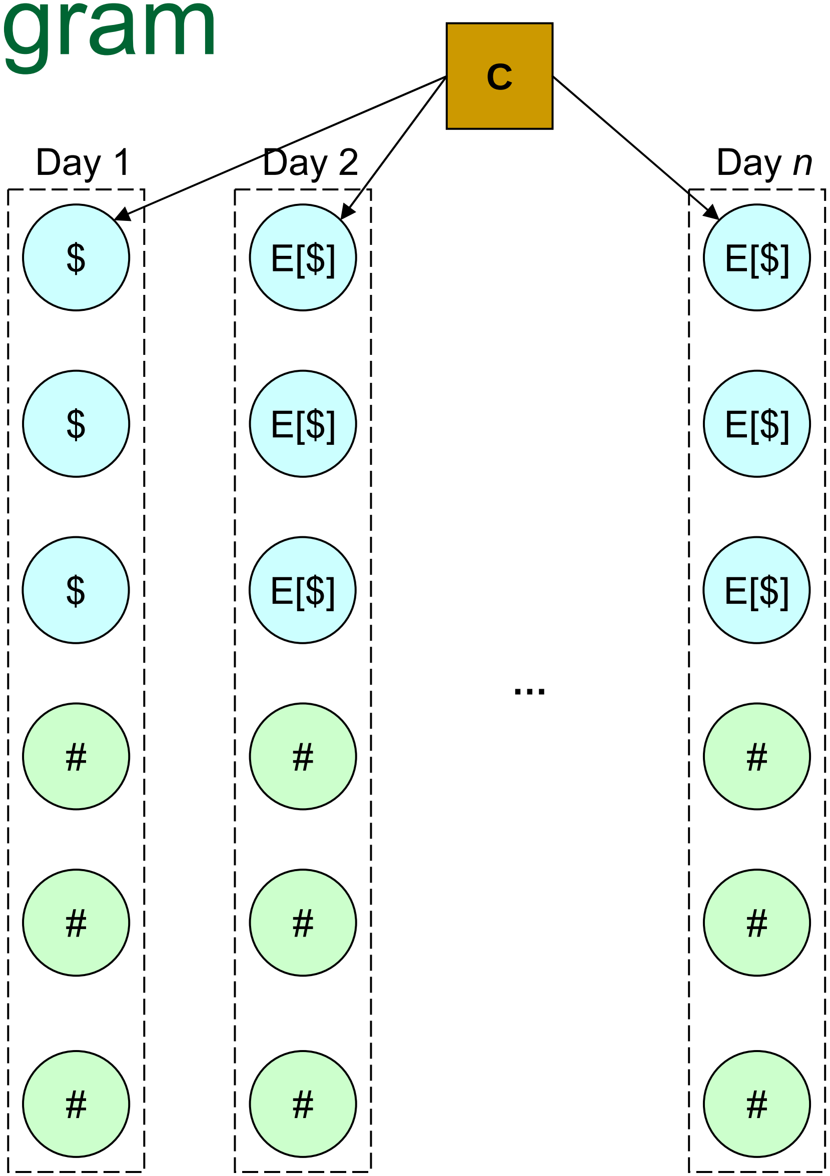
SILP Program



SILP Program

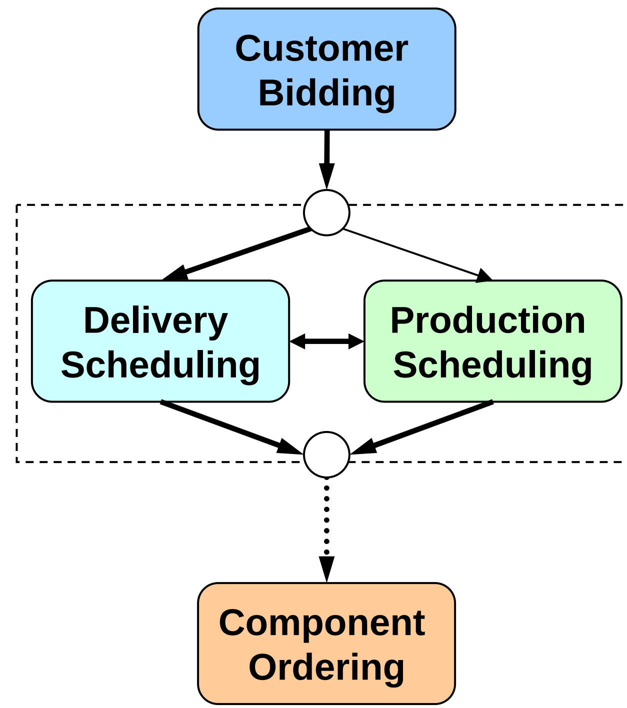


SILP Program

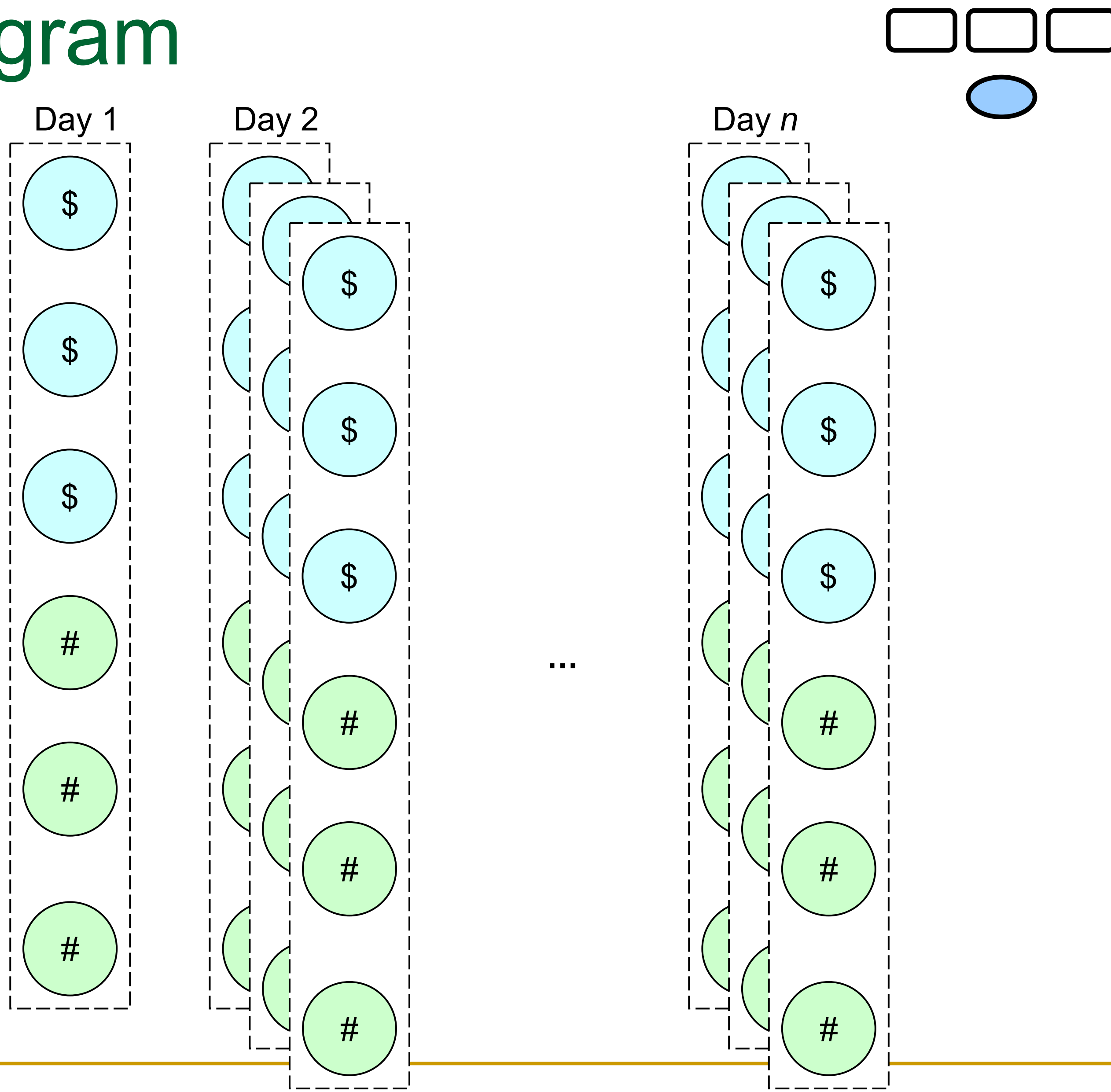


SAA

- **Sample Average Approximation (SAA)**
  - Shapiro *et al* 2001
    - Benisch *et al* 2004
  - Delivery and Production Scheduling
    - ILP that maximizes expected profit
    - *n*-day horizon
    - *k*-samples
      - Drawn from uncertainty distribution



SAA Program



Outline

- Introduction to Problem
- Model
  - Customer Market Process
  - Component Market Process
- **Application: Scheduling**
  - Agents
  - **Experiments**
- Conclusions

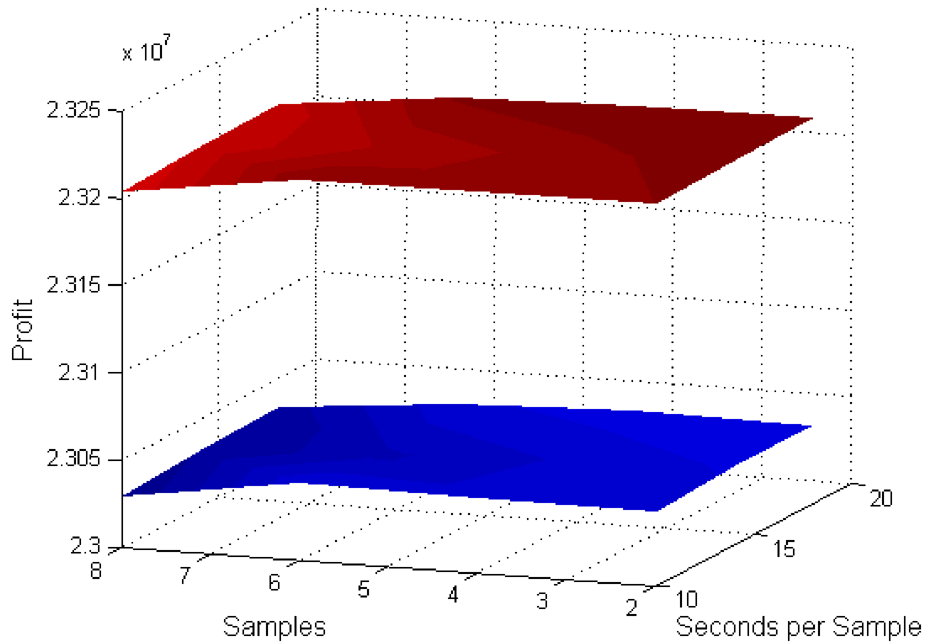
Common Test Setup

- 11-computer cluster
- ILPs solved with CPLEX 10.1
  - Told to emphasize feasibility over optimality
- Used profit as a measure of solution quality
  - Revenue less late penalties and storage costs

Experiment 1: SAA

- **Question:** Given a global time cap, does it make more sense for SAA to quickly consider more samples, or spend more time optimizing fewer samples?
  - 2, 4, 6, or 8 sample
  - 10, 14, or 18 seconds per sample
- For each combination ran 100 simulations
  - 30-days of simulated steady-state behaviour

Experiment 1: SAA (Results)



- Flat surface
  - Neither dimension significant for configurations that could be reasonably solved in TAC

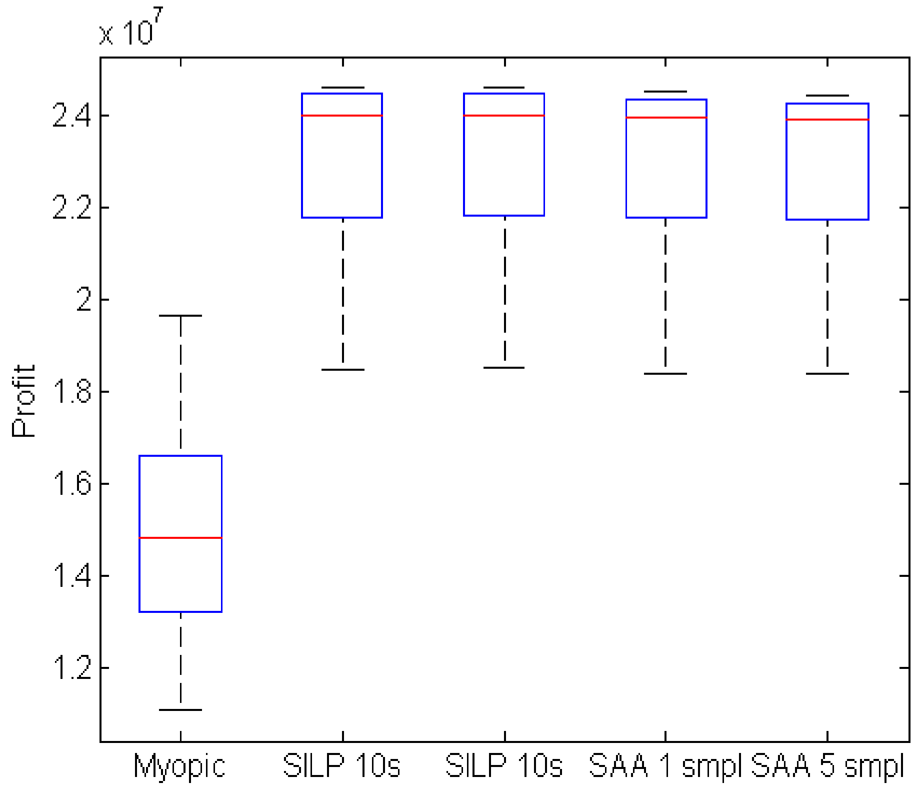
Experiment 2: Algorithm Comparisons

- **Question:** Given these three algorithms and a time constraint, which algorithm should one use?
  - Myopic
  - 2-day SILP with 10s cap
  - 2-day SILP with 50s cap
  - 3-day SAA with 1 sample, 10s cap
  - 3-day SAA with 5 sample, 50s cap

- For each algorithm ran 100 simulations
  - 30-days of simulated steady-state behaviour
- CPLEX solved the Myopic ILP in under 10s

Experiment 2: Algorithm Comparisons (Results)

- SILP and SAA beat Myopic
- SILP and SAA not significantly different
- Altering time cap makes no significant difference



Outline

- Introduction to Problem
- Model
  - Customer Market Process
  - Component Market Process
- Application: Scheduling
  - Agents
  - Experiments
- **Conclusions**

Conclusions

- From experiments
  - SILP and SAA were not significantly different for examined configurations
  - Increasing the number of samples in time-constrained SAA optimization did not significantly increase profit
  - Early approximations were usually quite good

Conclusions

- From testing approach and framework
  - Easy to set up and run large experiments
    - 1200 simulation in the first experiment
    - 500 simulations in the second
  - Simple to parallelize
  - More control over parameters
    - Time cap and simulation length altered
  - Accurate model of Customer Market Process
    - Game data likely given model
    - Low prediction error

Future Directions

- Data generated component market model
- Improve our model of the customer market
  - Priors during EM parameter estimation
  - Larger data set
  - TAC SCM Prediction Challenge
- Expand set of metrics
- Use framework integrate component ordering and customer bidding

Thank You

Questions and comments