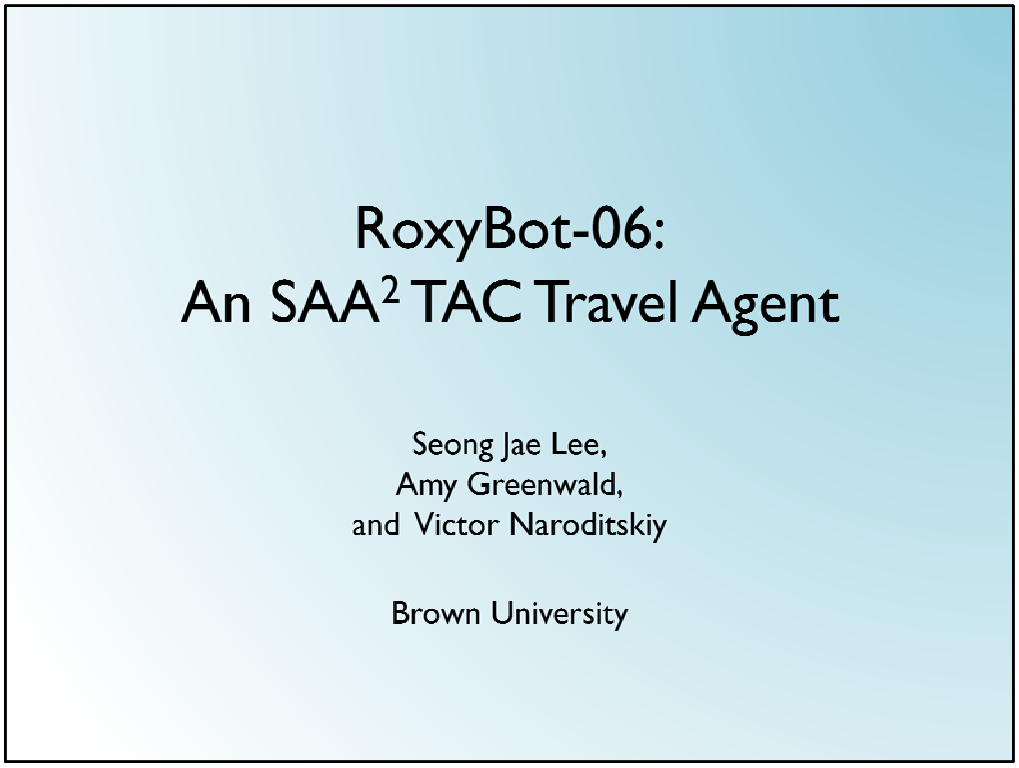


RoxyBot-06: An SAA^2 TAC Travel Agent

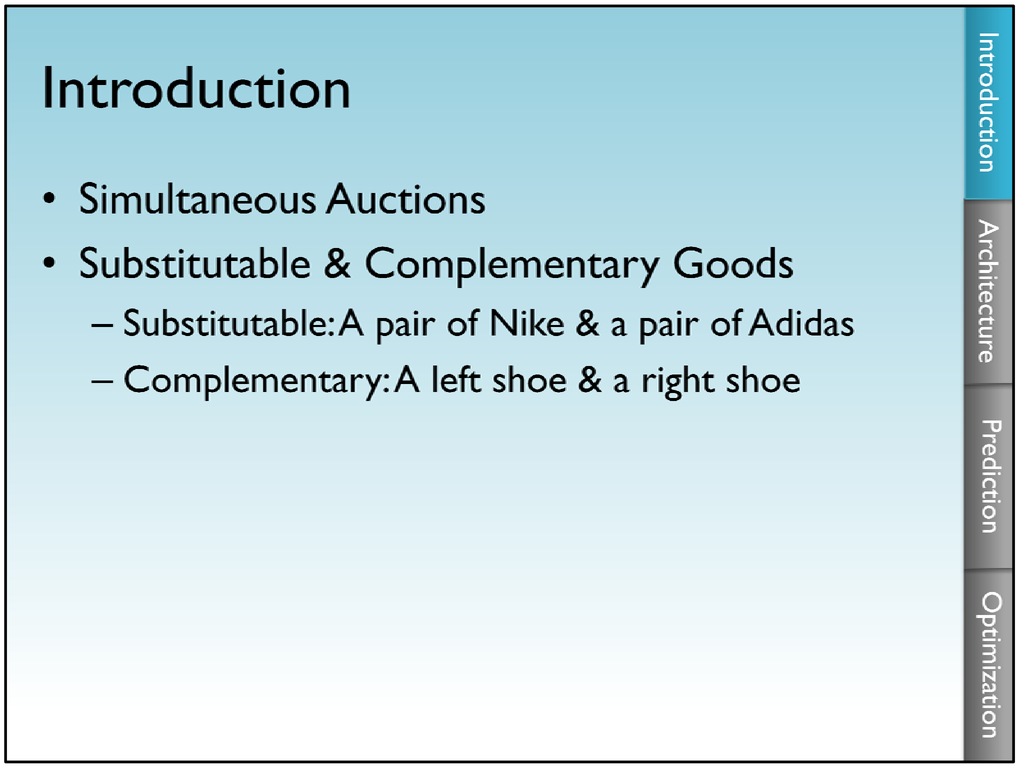
Seong Jae Lee,
Amy Greenwald,
and Victor Naroditskiy

Brown University

Hi, I am Seong Jae Lee.
I am going to present the winning agent of Trading Agent Competition, or TAC, 2006.
This paper is a joint work with Amy Greenwald and Victor Naroditskiy.

Our agent, RoxyBot is based on two distinctive features:
Simultaneous Ascending Auction and Sample Average Approximation,
hence making our paper title SAA squared, from both acronyms.

Although our algorithm is designed for TAC,
I believe that our agent's extensible features will lead to
efficient algorithms for simultaneous auctions,
which is common in e-commerce such as amazon.com or e-bay.com.



Introduction

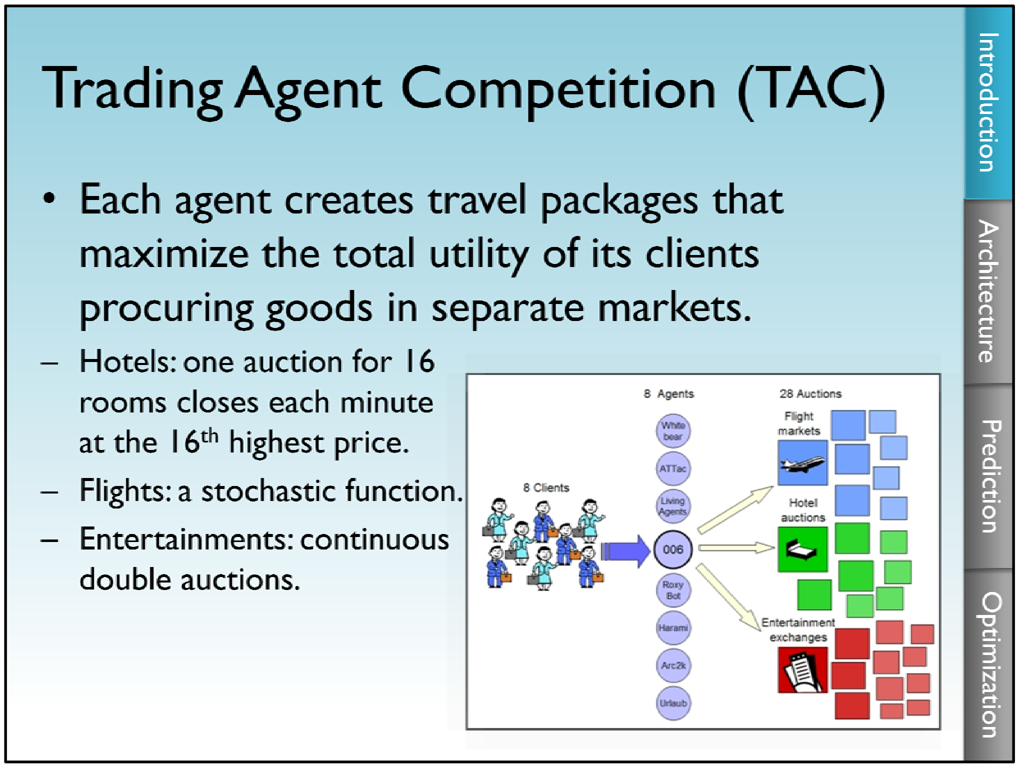
- Simultaneous Auctions
- Substitutable & Complementary Goods
  - Substitutable: A pair of Nike & a pair of Adidas
  - Complementary: A left shoe & a right shoe

Introduction
Architecture
Prediction
Optimization

Simultaneous auctions give a challenge to bidders,
particularly when substitutable and complementary goods are on sale.

Substitutable goods have subadditive values, and
complementary goods have superadditive values.

For example, a pair of Nike shoes and a pair of Adidas shoes are substitutable,
because the bidder would not want both of them,
while a left shoe and a right shoe are complementary goods,
because the bidder would want both of them.



Trading Agent Competition (TAC)

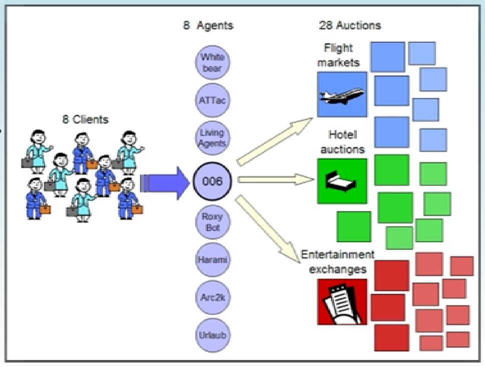
- Each agent creates travel packages that maximize the total utility of its clients procuring goods in separate markets.
  - Hotels: one auction for 16 rooms closes each minute at the 16th highest price.
  - Flights: a stochastic function.
  - Entertainments: continuous double auctions.

Introduction

Architecture

Prediction

Optimization



Trading Agent Competition is a virtual place market
designed to promote research on the trading agent problem.

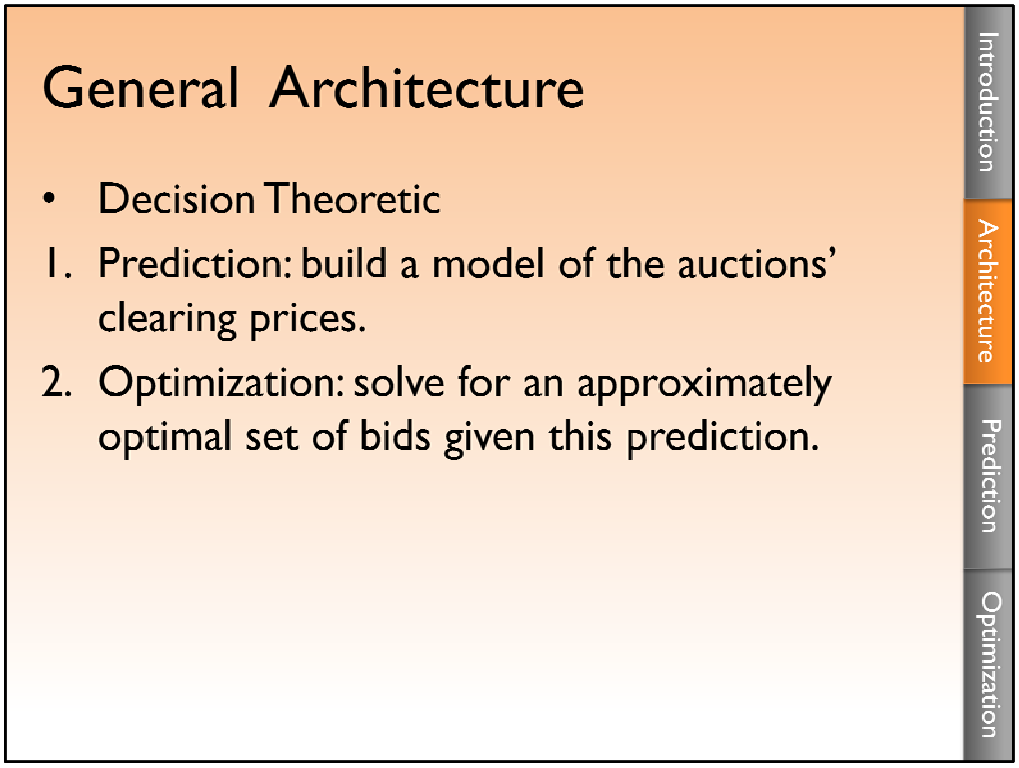
An agent's objective is
to create travel packages that maximize the total utility of its clients procuring goods in separate markets.

A travel package contains three kinds of goods, which is sold in different auction types.

Hotels are purchased from the TAC Seller.
One auction closes each minute in a random order.
When a hotel auction closes, 16 hotel rooms are sold with the 16th highest price.
There are two kinds of hotels at each day, making them substitutable goods.

Flight tickets are also purchased from the TAC Seller.
The posted-price is updated based on a stochastic function of time.
One can buy flights any amount at any moment.

Entertainment tickets are purchased from other agents in continuous double auctions.



General Architecture

- Decision Theoretic

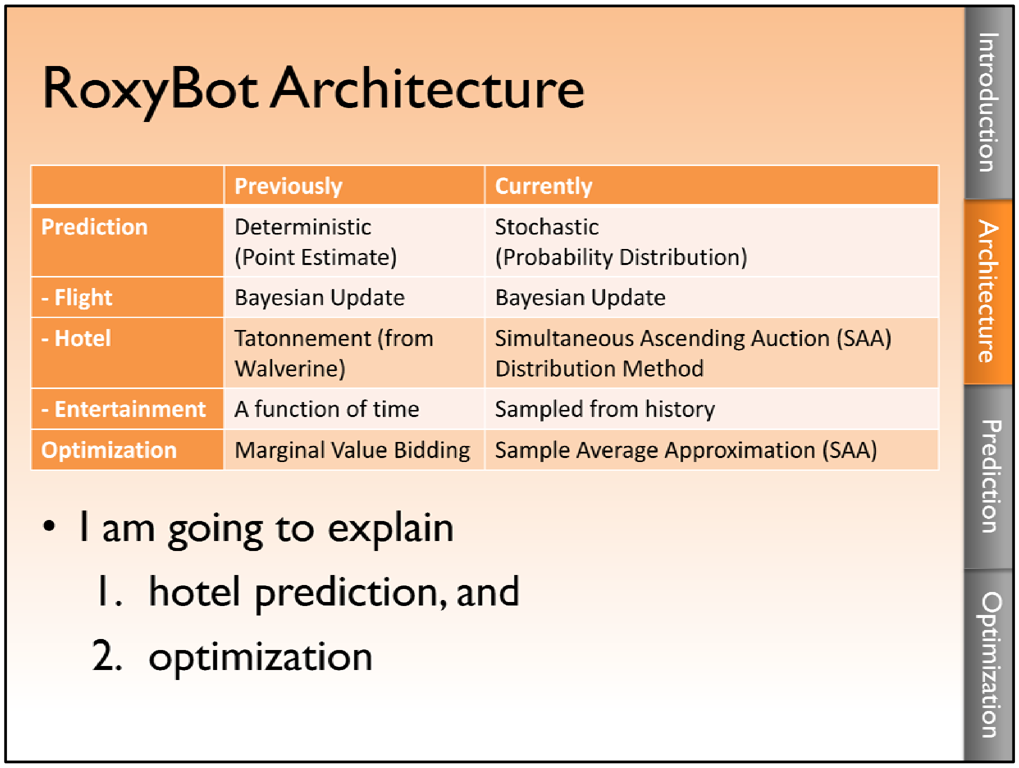
1. Prediction: build a model of the auctions' clearing prices.
2. Optimization: solve for an approximately optimal set of bids given this prediction.

Introduction
Architecture
Prediction
Optimization

Now, I am going to explain the general architecture of Roxybot.

At a high-level, the design of many successful TAC agents can be summarized as:
1. prediction: build a model of the auctions' clearing prices
2. optimization: solve for an approximately optimal set of bids, given this model.

In this case, the agent is playing with a decision theoretic approach:
it believes that prediction is accurate, and its bidding policy does not affect the clearing prices.



RoxyBot Architecture

	Previously	Currently
Prediction	Deterministic (Point Estimate)	Stochastic (Probability Distribution)
- Flight	Bayesian Update	Bayesian Update
- Hotel	Tatonnement (from Walverine)	Simultaneous Ascending Auction (SAA) Distribution Method
- Entertainment	A function of time	Sampled from history
Optimization	Marginal Value Bidding	Sample Average Approximation (SAA)

- I am going to explain
  1. hotel prediction, and
  2. optimization

Here is the comparison between previous RoxyBot and current Roxybot. Except the flight price prediction, everything is changed.

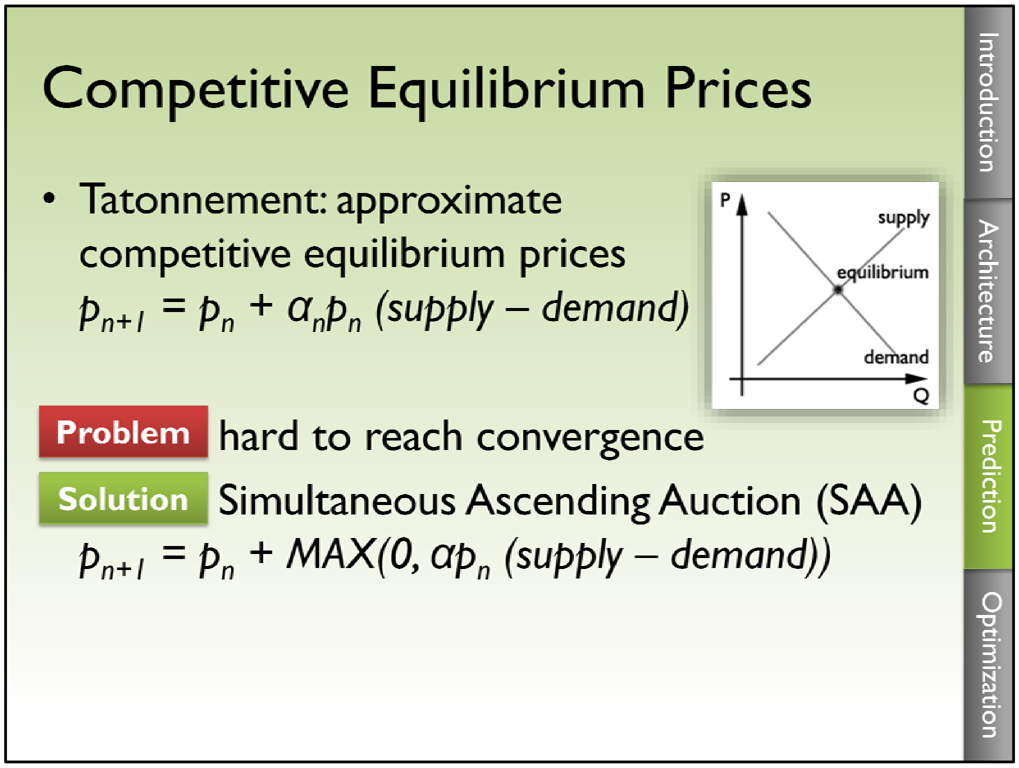
First, the price prediction is changed from point estimate to probability distribution. Instead of generating one prediction, we are generating several scenarios.

Second, for hotel price prediction, we made several changes from the implementation of tatonnement method.

Third, for entertainment ticket price prediction, we are sampling data from game history instead of making a point estimate prediction which is just a function of time.

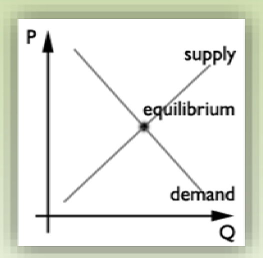
Finally, the optimization technique is changed from marginal value bidding to sample average approximation, which I will explain later.

It is a complicated process to speak all the innovations we made, so I am going to explain two major changes:
first, hotel prediction with Simultaneous Ascending Auction technique, and
second, Sample Average Approximation as an optimization technique.



Competitive Equilibrium Prices

- Tatonnement: approximate competitive equilibrium prices
  $p_{n+1} = p_n + \alpha_n p_n\,(supply - demand)$



Problem hard to reach convergence

Solution Simultaneous Ascending Auction (SAA)

$p_{n+1} = p_n + MAX(0, \alpha p_n\,(supply - demand))$

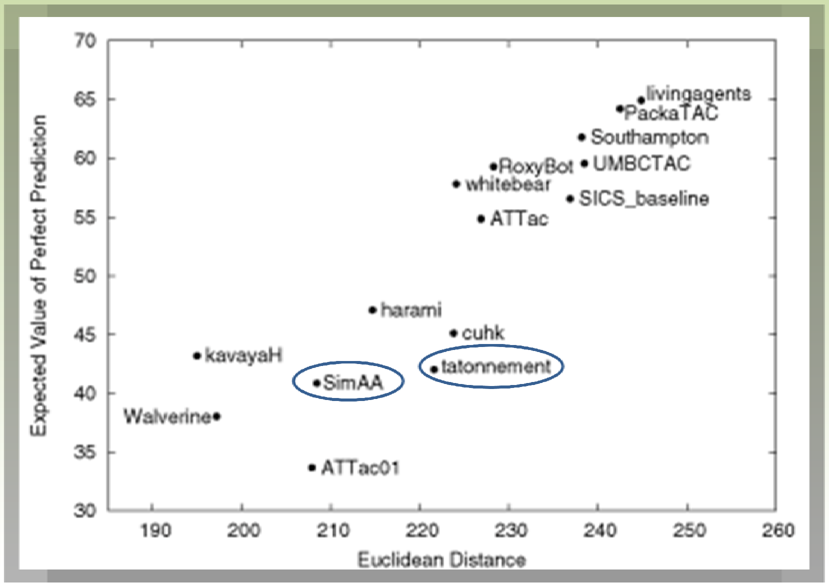
We wanted to approximate competitive equilibrium prices.
To get it, we modified the tatonnement method used in Walverine, a TAC Agent of University of Michigan.

Walverine's tatonnement process updates the prices by excessive demand for each iteration. Due to time constraint and a computer's computational limit, Walverine forced convergence. To do it, Walverine decreased alpha value over time, which is multiplied to the excessive demand.

But, the prediction was sometimes far from the mathematical convergence in some extreme cases. Moreover, our implementation of tatonnement process took a lot of time. So we changed the tatonnement process so that the prices can never decrease, which ensures convergence within a small number of iterations. The modified procedure is called Simultaneous Ascending Auction (SAA).

In our implementation, predictions using Simultaneous Ascending Auction took a tenth of the time for predictions using the tatonnement process, and more than 99 percent of our predictions reached convergence. Plus, this approach is much more like the actual game setting, because in TAC, the hotel prices can only increase.

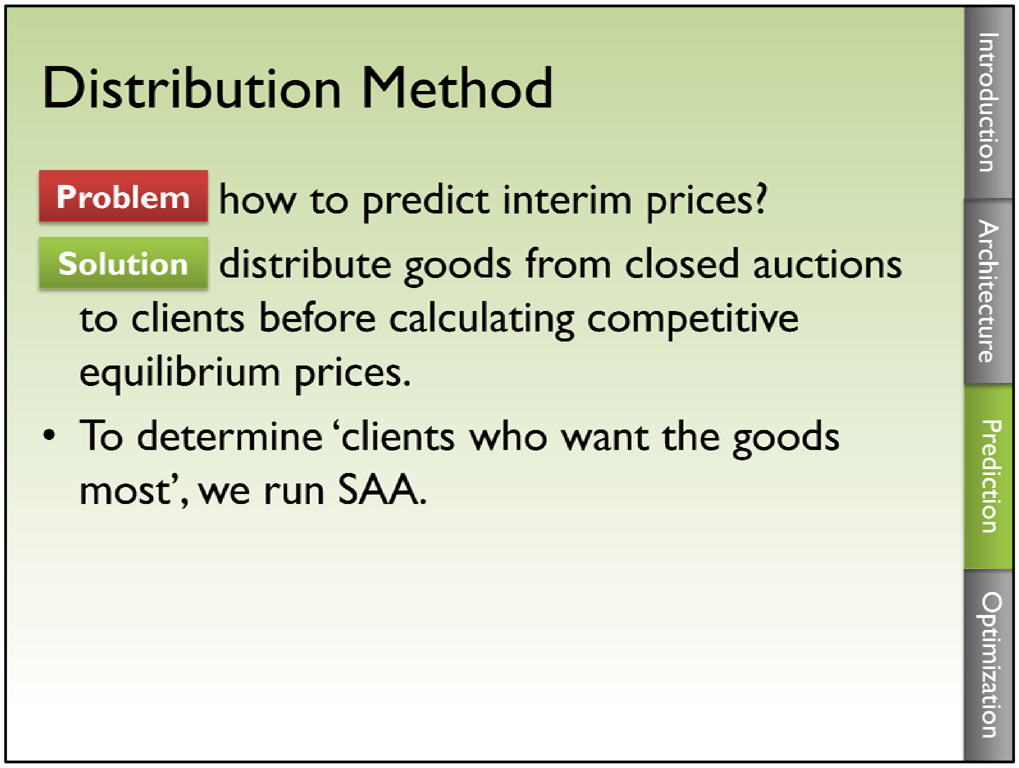
Performance



This is the Euclidian distance and the Expected Value of Perfect Prediction of TAC 2002. Euclidean Distance is the difference between the actual closing prices and prediction. The Value of Perfect Prediction is the difference between the value you would get if you predicted prices perfectly and the value you actually got.

The evaluation methods and the graph are from the price prediction in the Trading Agent Competition paper by University of Michigan. They collected each game's clearing prices and agents' predictions in TAC to generate this graph.

We plotted 'Tatonnement' and 'SAA' next to the agents who actually played in Trading Agent Competition. As you can see, our implementations showed fairly good scores. We can actually see that SAA performs better than Tatonnement; it is not only more faster, but also more accurate.



Distribution Method

Problem how to predict interim prices?

Solution distribute goods from closed auctions to clients before calculating competitive equilibrium prices.

- To determine 'clients who want the goods most', we run SAA.

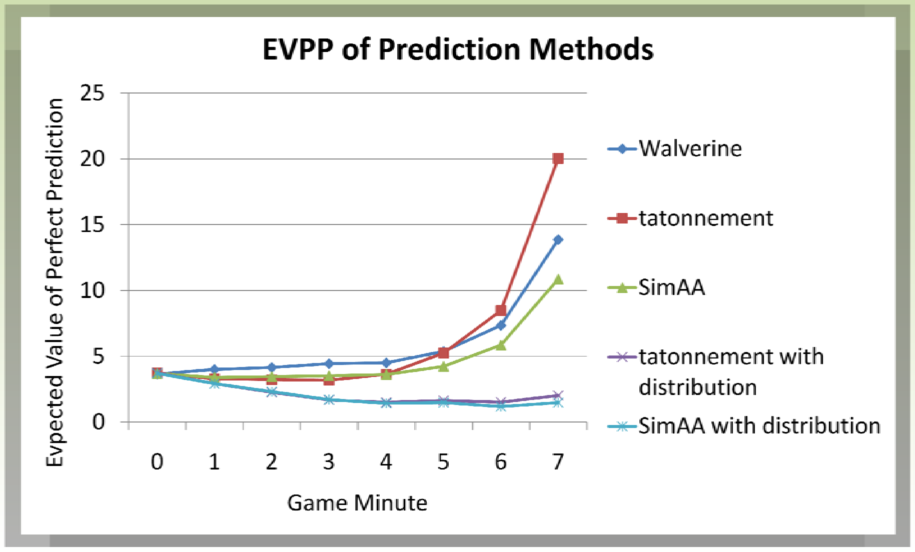
Now, I am going to describe for prediction method for interim prices.
Since the hotel auction closes at each minute, the hotel prices increase as the time passes.

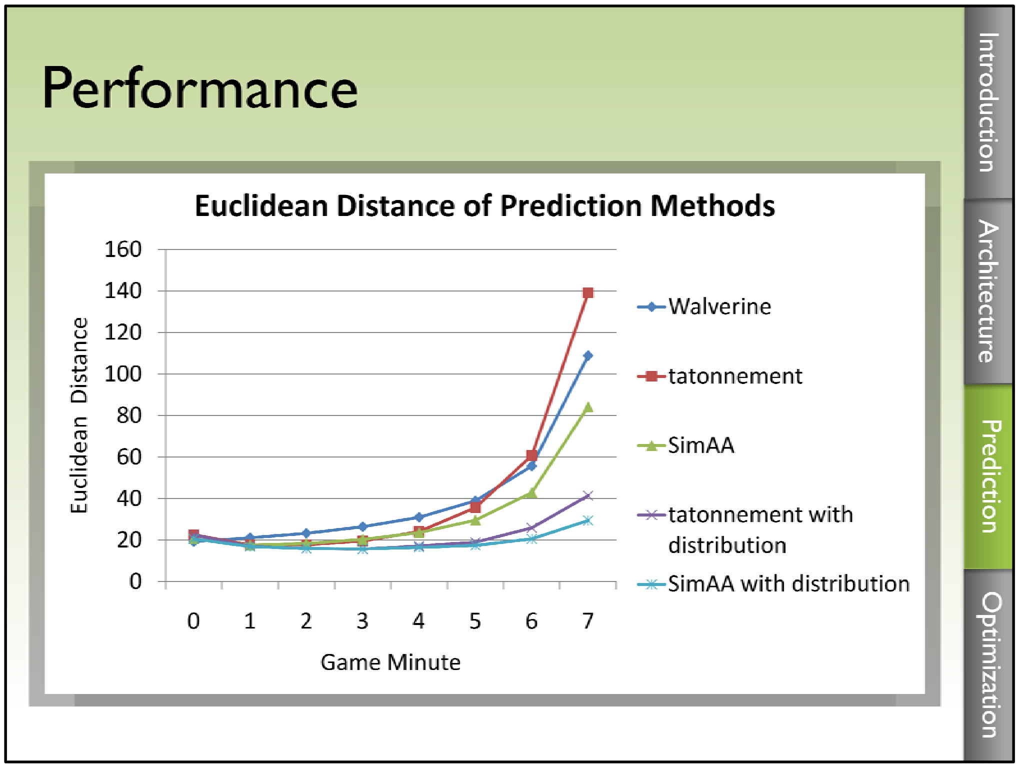
This happens because
first, once an agent wins some goods, it sticks to its complementary goods, and second, once an agent loses some goods, it sticks to its substitutable goods.

Thus, we cannot just use our method as it is.

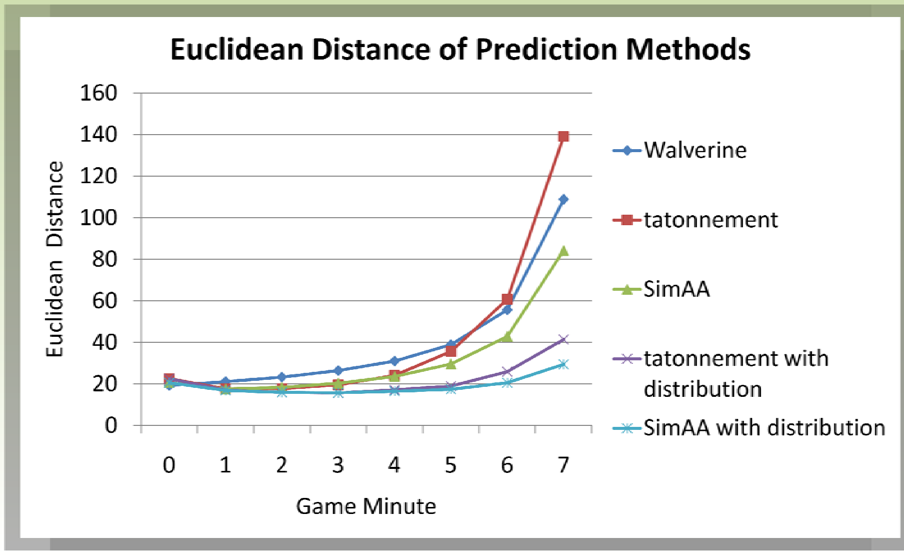
Because the increase of hotel prices occurs due to goods they already won, RoxyBot also simulates this tendency by distributing goods from closed auctions to clients used in SAA process.
We distributed hotel rooms to those who wanted the goods in the equilibrium state, to select 16 clients who wanted them most.
Of course, this equilibrium is also determined by the SAA process.
Thus, we are running SAA process twice per prediction,
which takes roughly twice much time of a normal SAA procedure.

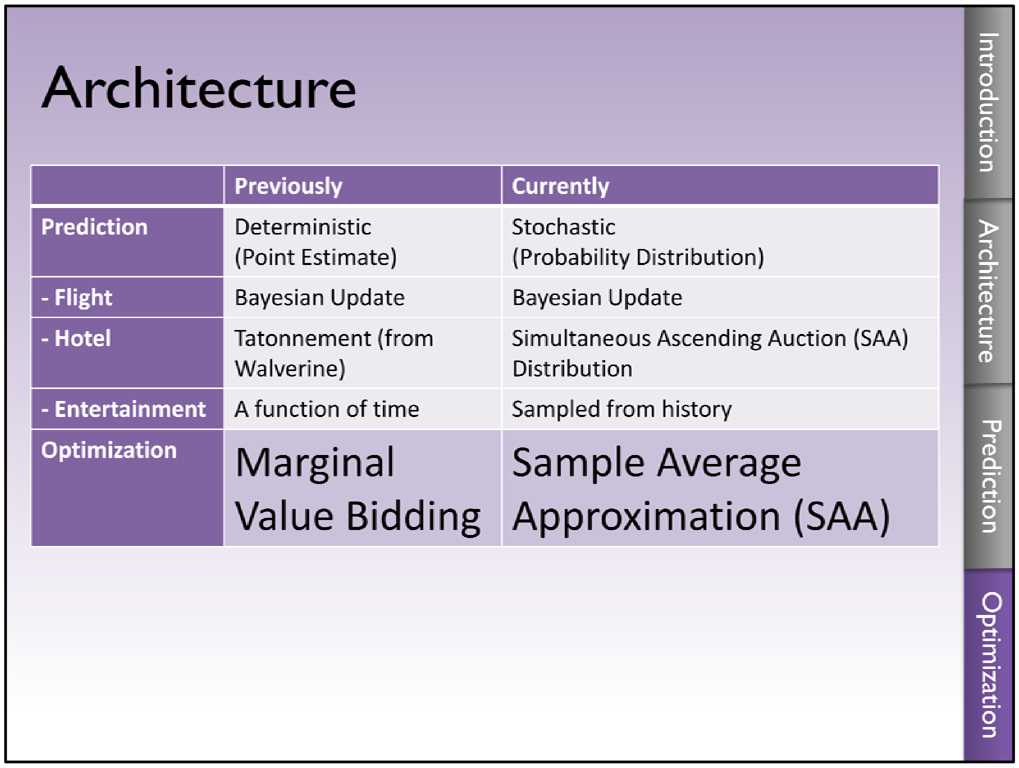
Performance





Performance

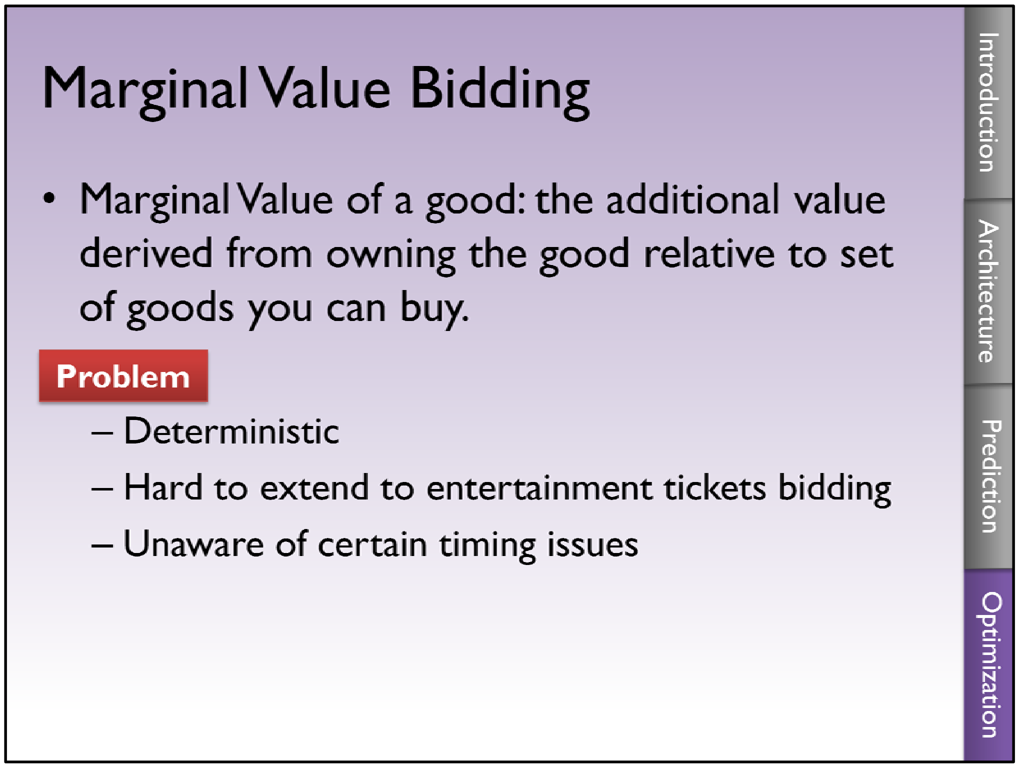




Architecture

	Previously	Currently
Prediction	Deterministic (Point Estimate)	Stochastic (Probability Distribution)
- Flight	Bayesian Update	Bayesian Update
- Hotel	Tatonnement (from Walverine)	Simultaneous Ascending Auction (SAA) Distribution
- Entertainment	A function of time	Sampled from history
Optimization	Marginal Value Bidding	Sample Average Approximation (SAA)

Next, I am going to explain optimization techniques; marginal value bidding and sample average approximation.



Marginal Value Bidding

- Marginal Value of a good: the additional value derived from owning the good relative to set of goods you can buy.

Problem

- – Deterministic
- – Hard to extend to entertainment tickets bidding
- – Unaware of certain timing issues

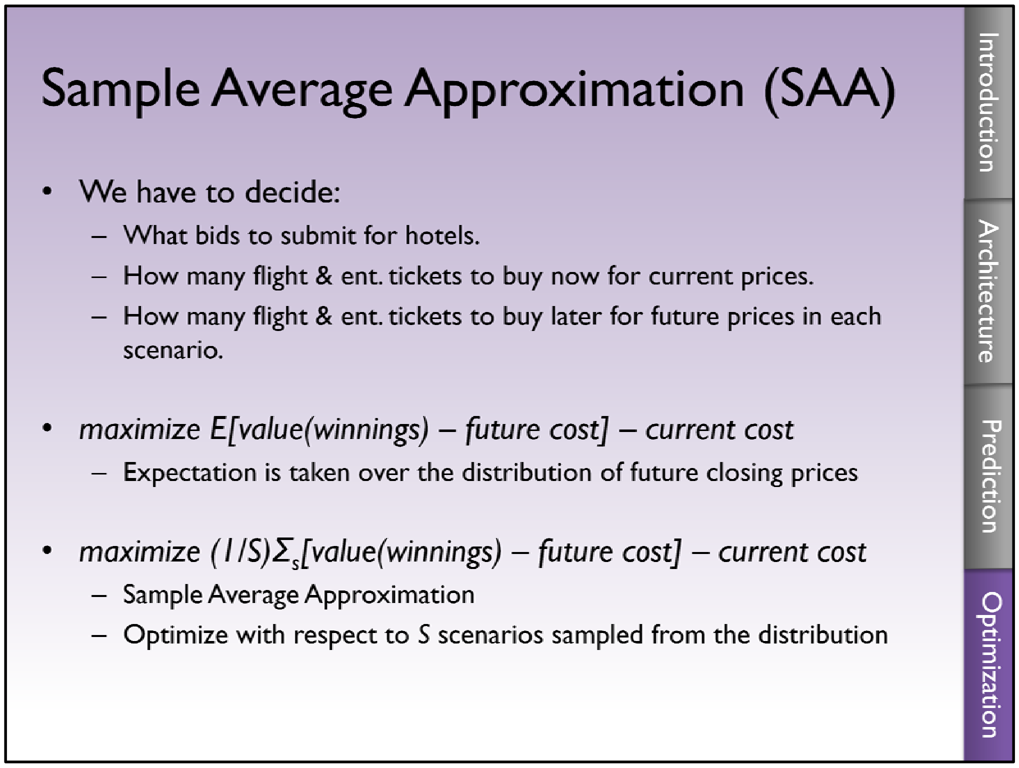
Introduction
Architecture
Prediction
Optimization

Many successful TAC agents including our previous agents used marginal value to as a bidding price.
The marginal value of a good is the additional value derived from owning the good relative to set of goods you can buy.

But we have several problems on the marginal value bidding policy.
First, this policy does not have an intuitive way to use stochastic prediction;
it is designed for the point-estimate prediction.

Second, this policy is hard to extend to the entertainment bidding.
Marginal Value is same as the maximum price one can afford, and
because entertainment tickets is purchased in a double auction, one should not bid as much as one can afford.
Thus, our previous model required bid shading, which was… sort of hacky.

Third, the agent is not aware of timing.
Since flight and entertainment tickets can be purchased continuously, timing is important.
For example, we can postpone purchasing flight tickets after we purchase hotels, assuming it is possible to lose hotels.



Sample Average Approximation (SAA)

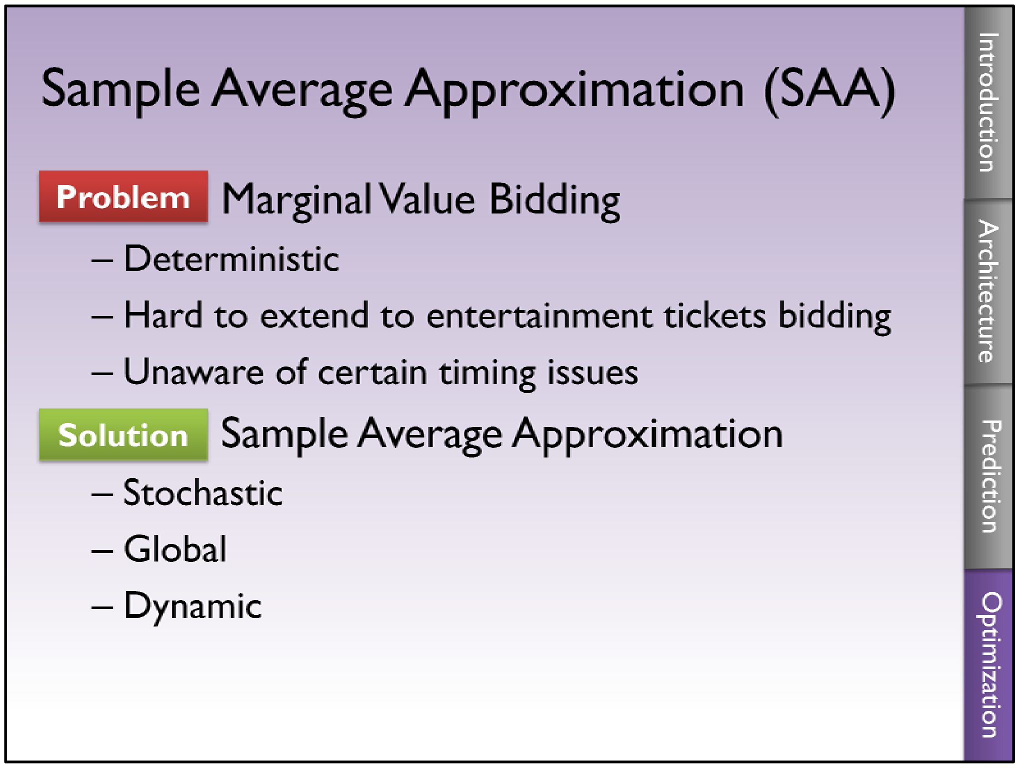
- We have to decide:
  - What bids to submit for hotels.
  - How many flight & ent. tickets to buy now for current prices.
  - How many flight & ent. tickets to buy later for future prices in each scenario.
- *maximize E[value(winnings) – future cost] – current cost*
  - Expectation is taken over the distribution of future closing prices
- *maximize* $(1/S)\Sigma_s$*[value(winnings) – future cost] – current cost*
  - Sample Average Approximation
  - Optimize with respect to *S* scenarios sampled from the distribution

In sample average approximation process we decides,
first, what bids to submit for hotels,
second, how many tickets to buy now for current prices, and
third, how many tickets to buy later for future prices in each scenarios.

This is done by finding the set of bids that maximizes,
the expected value of winnings minus future cost, minus current cost,
while the expectation is taken over the distribution of future closing prices.

In Sample Average Approximation, we optimize with respect to S number of scenarios sampled from the distribution.

We accomplished this process using CPLEX, and its integer linear programming equation is described at the end of our paper.



Sample Average Approximation (SAA)

Problem Marginal Value Bidding

- Deterministic
- Hard to extend to entertainment tickets bidding
- Unaware of certain timing issues

Solution Sample Average Approximation

- Stochastic
- Global
- Dynamic

Introduction
Architecture
Prediction
Optimization

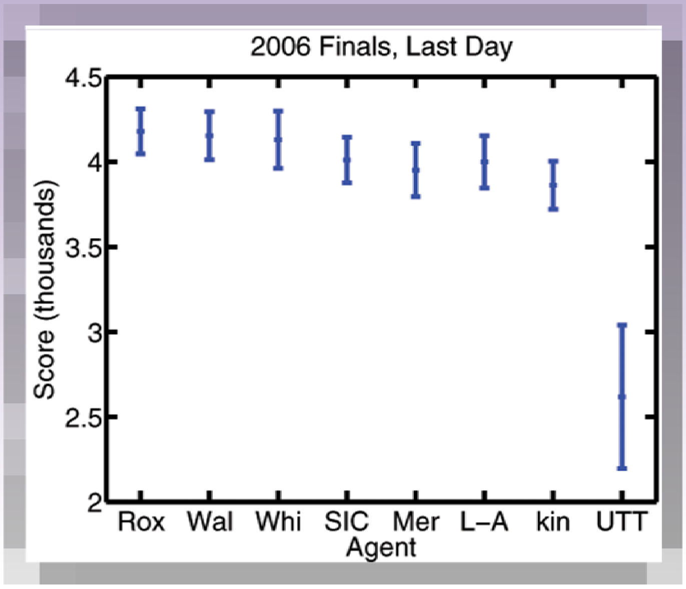
We can see SAA solves previous problems.

First, it uses stochastic predictions.

Second, it is global; it simultaneously considers flight, hotel, and entertainment bids in unison. We can easily extend different types of auctions.

Finally, it is dynamic; it simultaneously reasons about bids to be placed in both current and future stages of the game.

Performance



This is the score of TAC 2006 Final with 95% confidence intervals.
It does not show that our agent is doing absolutely better than other agents,
but I think this result is good enough to verify the performance of our algorithm.