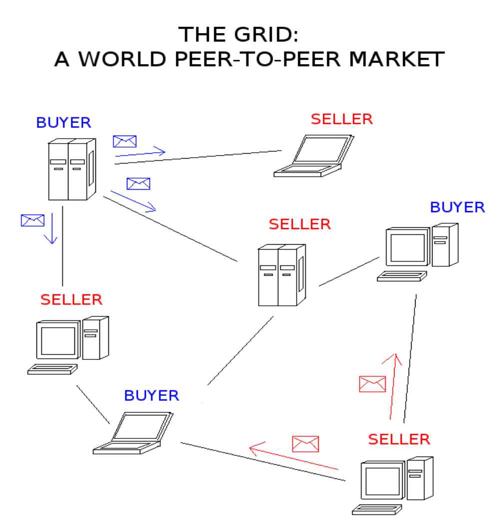
A Markovian Futures Market for Computing Power

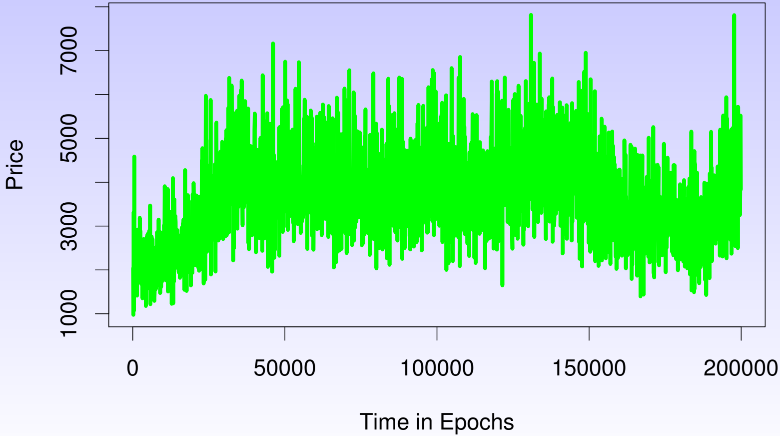
Fernando Martinez Peter Harrison Uli Harder

A distributed economic solution: MaGoG



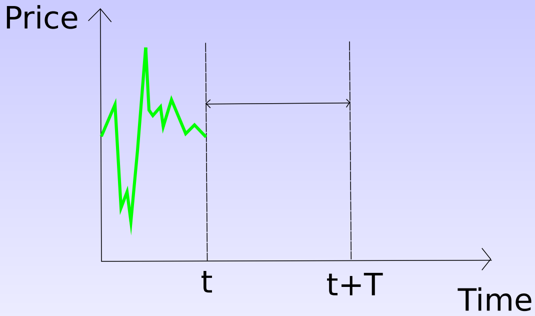
- A world peer-to-peer market
- No central auctioneer
- Messages are forwarded by neighbours, and a copy remains in their pubs
- Every node has a pub, a trading floor, where deals are closed

Spot price evolution



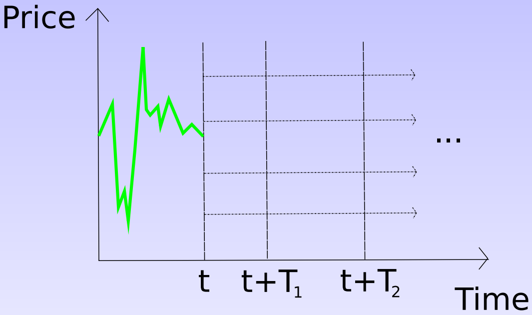
Future contracts

- Computing power is non-storable, and therefore non-tradeable



- Future contract: agreement to buy/sell something at a future date for a fixed price

Future contracts



- Futures allow to trade the underlying computing power
- They extend the trading spectrum, allowing maximization of the use of resources, as well as hedging and speculation

Future contracts

- Financials and storable commodities have a relation between the Spot price and the Future price: $F(t, T) = S(t)e^{rT}$

  t Present date

  T Remaining time to maturity date

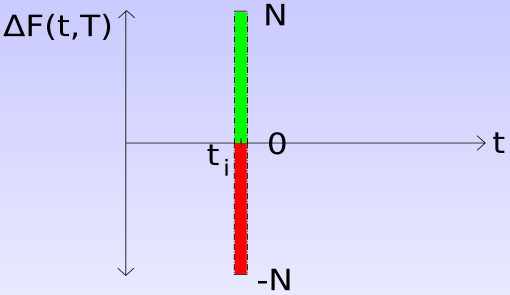
  r Interest rate

- Since computing power is non-storable: there is no direct relation between the Spot price and the Future price

$\Longrightarrow$ Model future prices directly

Future contracts

- We consider the variation in price, rather than the price itself:

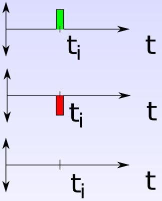


- We limit the price variation at a particular time step by the number of agents (finite)
- Both positive and negative variations are allowed

Future contracts

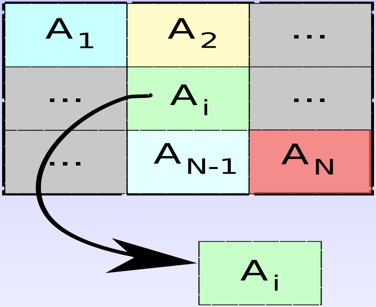
- We introduce the concept of *market pressure*, which determines the price variation of the market:

Market pressure		Price variation
Demand > Supply	⟶	$\Delta F(t,T)$
Demand < Supply	⟶	$\Delta F(t,T)$
Demand = Supply	⟶	$\Delta F(t,T)$



Future contracts

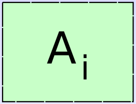
- The market is formed by its market participants



- We therefore specify the behaviour of each agent

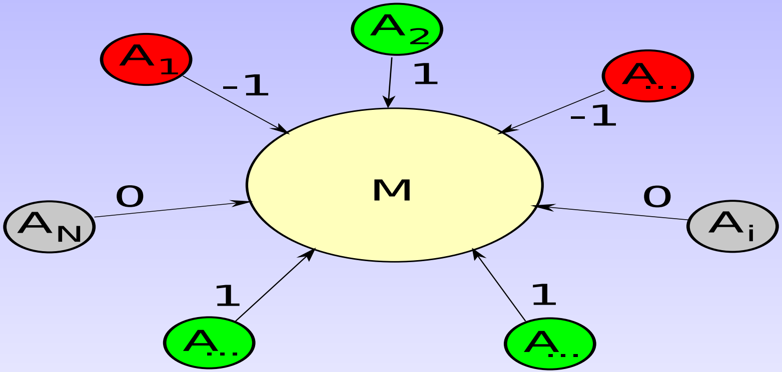
Future contracts

- Agents take Markovian decisions
- Discrete-time Markov chain. Independent for each agent.
- Three possible actions or states: $\{-1, 0, 1\}$



$$T_i = \begin{pmatrix} T_i(-1,-1) & T_i(-1,0) & T_i(-1,1) \\ T_i(0,-1) & T_i(0,0) & T_i(0,1) \\ T_i(1,-1) & T_i(1,0) & T_i(1,1) \end{pmatrix}$$

Future contracts



- Agents trade future contracts of computing power for delivery at an arbitrary future date
- Agents submit ‘market orders’ with their intention to buy(1), sell(-1) or hold(0) at the current market price

Market model

- The market state is the result of considering the individual actions of all agents $\longrightarrow 3^N$ states!!
- By using the concept of market pressure:
  Market state = Sum of the individual states of the agents
- Then the number of market states is reduced to $2N + 1$

Transition probability matrix of the market

$$M = (m_{sd} \mid -N \leq s, d \leq N),$$

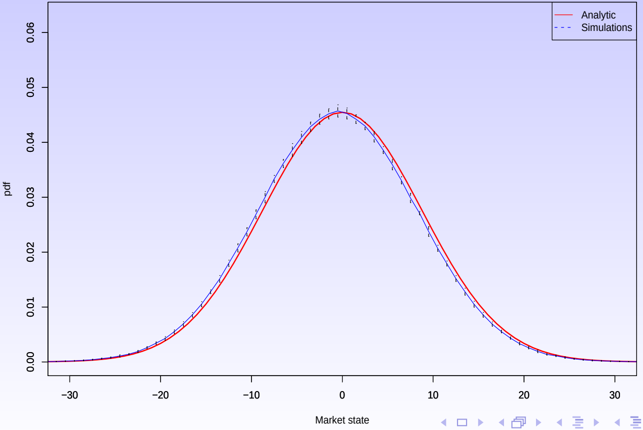
$$M = \begin{pmatrix} P(-N,-N) & \dots & P(-N,0) & \dots & P(-N,N) \\ \vdots & & \vdots & & \vdots \\ P(0,-N) & \dots & P(0,0) & \dots & P(0,N) \\ \vdots & & \vdots & & \vdots \\ P(N,-N) & \dots & P(N,0) & \dots & P(N,N) \end{pmatrix}$$

Transition probability matrix of the market

- The global matrix is calculated via generating functions, which use convolutions to generate all the states
- Calculations are simplified when all agents are equal
- Normally there will be a few groups of agents, each group containing the same kind of agents

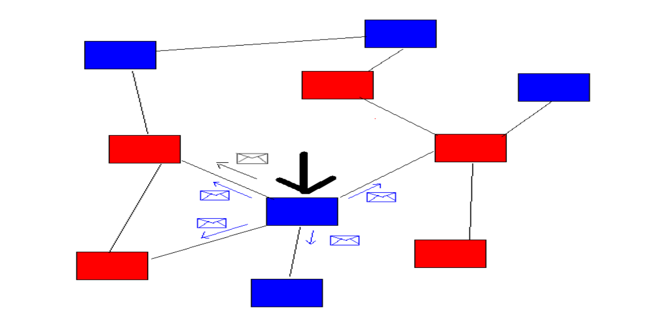
Simulation

- An ideal simulation setup with a fully connected network gives the same results as the analytic model



Simulation

- For a non-ideal simulation setup (peer-to-peer), shifting and scaling factors need to be found to design the architecture accordingly

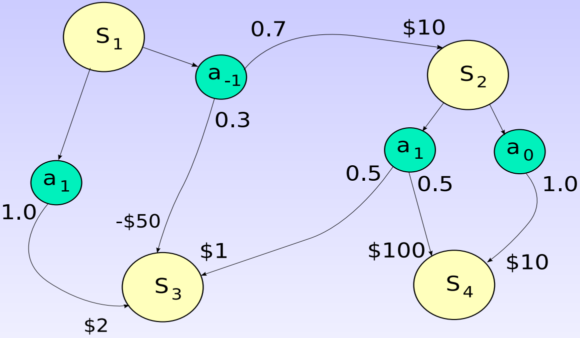


Futures trading

- Allow trading the underlying computing power
- Maximise use of resources
- Hedging
- Speculation

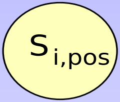
MDP

- Markov Decision Processes are used for decision-making in sequential, uncertain environments



- The decision maker receives a reward depending on his chosen action and the change in the system state

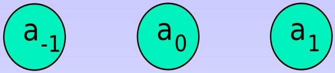
MDP: States of the system



$$S_{i,pos} = (i, |i|, pos) \quad \text{with} \quad i, pos \in \mathbb{Z} \cap [-N, N]$$

- i Price variation: given by the transition probability matrix of the market
- $|i|$ Trading volume: available to be bought or sold
- pos Open position of the trader

MDP: Actions of the trader



- The trader can buy one future contract (1), sell one future contract (-1) or hold his position (0) at every decision epoch
- He is limited to have an open position between $-N$ and N
- The number of decision epochs is infinite

MDP: Reward for the trader $

- The trader receives a reward depending on his actions and the evolution of the system
- We specify a reward that consists of two parts
- The first part is the profit/loss due to the trader's position and the price variation

$$r_1(s,a) = \sum_{j \in S} r_1(s,a,j) p(j|s,a)$$

$$r_1(s,a,j) = i_j * pos_j$$

MDP: Reward for the trader $

- The second part of the reward is a penalty for being unable to liquidate the open position

$$r_2(s,a) = \sum_{j \in S} r_2(s,a,j) p(j|s,a)$$

$$r_2(s,a,j) = -c * max(|pos_j| - |i|_j, 0), \quad c \in \mathbb{R}^+$$

- Total reward:

$$r(s,a) = r_1(s,a) + r_2(s,a)$$

MDP: Optimal trading policy

- Find an optimal trading policy
- Infinite number of decision epochs $\longrightarrow$ apply a discount factor λ $(0 \leq \lambda < 1)$ that makes future rewards less valuable
- Expected total present value of the reward:

$$v_\lambda^\pi(s) = E_s^\pi\{\sum_{t=1}^{\infty} \lambda^{t-1} r(X_t, Y_t)\}$$

$\Longrightarrow$ Find the policy π that maximizes this reward

MDP: Optimal trading policy via Linear Programming

- Easy formulation
- Discounted Markov Decision problem $\Longrightarrow$ Linear Programming problem

MDP: Optimal trading policy via Linear Programming

- choosing $\alpha(j), j \in S$ (being S the state space of the MDP) to be positive scalars with $\sum_{j \in S} \alpha(j) = 1$
- The dual linear program consists of maximizing:

$$\sum_{s \in S} \sum_{a \in Ac_s} r(s,a)x(s,a)$$

subject to:

$$\sum_{a \in Ac_j} x(j,a) - \sum_{s \in S} \sum_{a \in Ac_s} \lambda p(j|s,a)x(s,a) = \alpha(j)$$

and $x(s,a) \geq 0$ for $a \in Ac_s$ and $s \in S$.

MDP: Optimal trading policy via Linear Programming

- Solving the dual $\equiv$ finding the $x(s, a)$
- We then obtain a decision rule for each state by choosing the action that gives the highest probability:

$$P\{d_x(s) = a\} = \frac{x(s, a)}{\sum_{a' \in Ac_s} x(s, a')}$$

- The set of the decision rules for each state of the MDP forms the policy

MDP: Example

$$T_1 = \begin{pmatrix} 0.3 & 0.4 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.3 & 0.4 & 0.3 \end{pmatrix} \qquad T_2 = \begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.4 & 0.2 & 0.4 \\ 0.3 & 0.4 & 0.3 \end{pmatrix}$$

$$M = \begin{pmatrix} 0.120000 & 0.250000 & 0.330000 & 0.210000 & 0.090000 \\ 0.120000 & 0.238533 & 0.326178 & 0.213822 & 0.101467 \\ 0.111000 & 0.236000 & 0.329333 & 0.222667 & 0.101000 \\ 0.102222 & 0.231852 & 0.331852 & 0.231852 & 0.102222 \\ 0.090000 & 0.240000 & 0.340000 & 0.240000 & 0.090000 \end{pmatrix}$$

$$S_{i,pos} = (i, |i|, pos), \quad for \quad i, pos \in \mathbb{Z} \cap [-2, 2]$$

$$Ac_s = \{-1, 0, 1\}, \quad for \quad s \in S$$

MDP: Example

- Penaly factor $c = 0.1$
- Discount factor $\lambda = 0.95$
- The dual is solved with GLPK (GNU Linear Programming Kit), using the same value for all the $\alpha(j)$
- In particular, the standard LP solver of GLPK, *glpsol*, is used, and an optimal solution is found by the simplex method

MDP: Example

- Optimal policy: trader's optimal action for each state of the MDP

$S_{0,-2}$	1
$S_{0,-1}$	-1
$S_{0,0}$	-1
$S_{0,1}$	-1
$S_{0,2}$	-1
$S_{1,-2}$	0
$S_{1,-1}$	-1
$S_{1,0}$	-1
$S_{1,1}$	-1
$S_{1,2}$	-1

$S_{2,-2}$	0
$S_{2,-1}$	-1
$S_{2,0}$	-1
$S_{2,1}$	-1
$S_{2,2}$	-1
$S_{-2,-2}$	1
$S_{-2,-1}$	-1
$S_{-2,0}$	1
$S_{-2,1}$	0
$S_{-2,2}$	-1

$S_{-1,-2}$	0
$S_{-1,-1}$	-1
$S_{-1,0}$	1
$S_{-1,1}$	-1
$S_{-1,2}$	-1

Conclusion

- World market for computing power
- Markov character of the agents. Reduced state space of the market by using *market pressure*
- Trading of future contracts of computing power. Optimal policy for MDP
- Further work will consider implementation on a peer-to-peer network
- And agents with variable behaviour depending on their neighbours

Thank you

{fermaror@doc.ic.ac.uk}