Stochastic Economic Model Predictive Control of Complex Drinking Water Networks

Alberto Bemporad

(joint work with **Ajay Kumar Sampathirao** and **Pantelis Sopasakis**)

IMT INSTITUTE FOR ADVANCED STUDIES LUCCA

http://imtlucca.it/alberto.bemporad

Outline

- Stochastic Model Predictive Control (of large-scale systems)
- Algorithmic aspects (parallel stochastic optimization)
- Application to the drinking water network of Barcelona

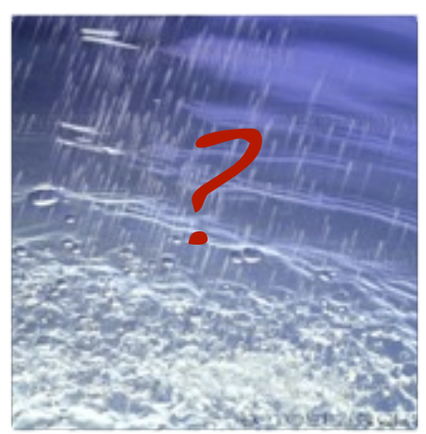
Stochastic Model Predictive Control

Optimize real-time decisions under uncertainty

- In many control problems decisions must be taken under **uncertainty**


renewable power

prices


water


demand

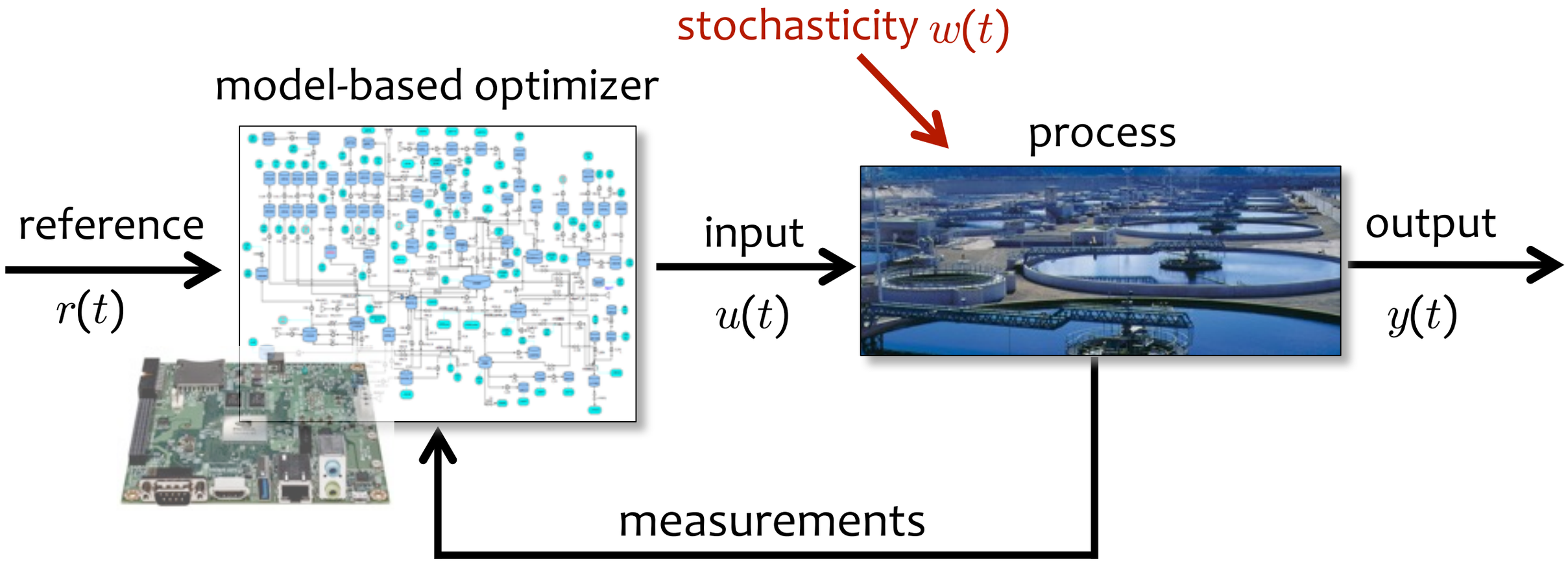

human (inter)action

- **Robust** control approaches do not model uncertainty (only assume that is bounded) and pessimistically consider the worst case

- **Stochastic models** provide instead additional information about uncertainty

- **Optimality** often sought (ex: minimize expected economic cost)

Stochastic Model Predictive Control (SMPC)



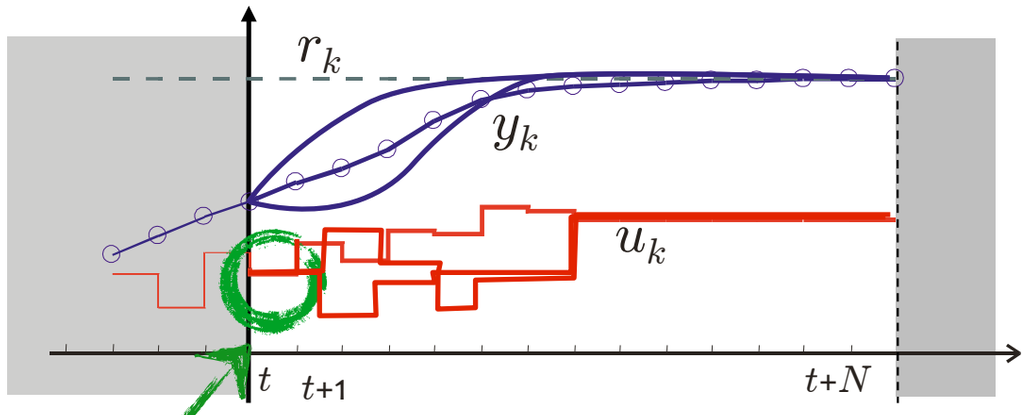
*Use a **stochastic** dynamical **model** of the process to **predict** its possible future evolutions and choose the "best" **control** action*

Stochastic Model Predictive Control

- At time t: solve a **stochastic optimal control** problem over a finite future horizon of N steps:

$$\min_{u} \quad E_w\left[\sum_{k=0}^{N-1} \ell(y_k, u_k, w_k)\right]$$

s.t.
$$x_{k+1} = A(w_k)x_k + B(w_k)u_k + f(w_k)$$
$$y_k = C(w_k)x_k + D(w_k)u_k + g(w_k)$$
$$u_{\min} \le u_k \le u_{\max}$$
$$y_{\min} \le y_k \le y_{\max}, \quad \forall w \quad \text{robustness}$$
$$x_0 = x(t) \quad \text{feedback}$$



$x(t)$ = process state

$u(t)$ = manipulated vars

$y(t)$ = controlled output

$w(t)$ = **stochastic disturbances**

- Solve stochastic optimal control problem w.r.t. future input sequence
- Apply the first optimal move $u(t)=u_0^*$, throw the rest of the sequence away
- At time $t+1$: Get new measurements, repeat the optimization. And so on …

Used in process industries since the 80's. Now spreading in other domains

Linear stochastic MPC w/ discrete disturbance

- **Linear stochastic** prediction model

$$\begin{cases} x_{k+1} = A(w_k)x_k + B(w_k)u_k + f(w_k) \\ y_k = C(w_k)x_k + D(w_k)u_k + g(w_k) \end{cases}$$

(A,B,C,D) are can be sparse (ex: network of interacting subsystems)

- **Discrete disturbance** $w_k \in \{w^1, \dots, w^s\}$ $\quad p_j = \mathbf{Pr}[w_k = w^j]$

$$p_j \geq 0, \ \sum_{j=1}^{s} p_j = 1$$

Often w_k is low-dimensional (ex: electricity price, weather, etc.)

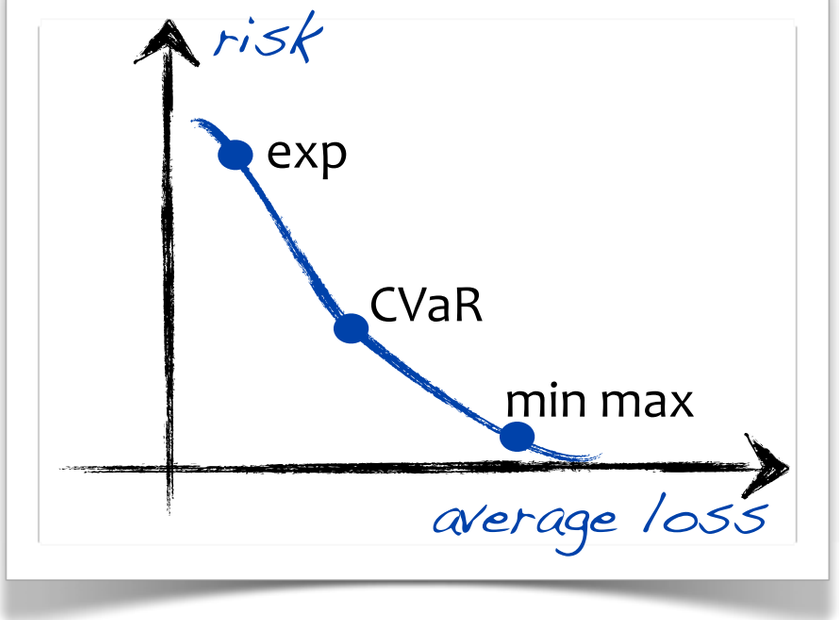
Probabilities p_j can have their own dynamics over time (ex: Markov chain) and can be **estimated from historical data**

Cost functions for SMPC to minimize

performance → $J(u,w) = \sum_{k=0}^{N-1} \ell(y_k, u_k, w_k)$

- **Expected performance**

$$\min_u E_w\left[J(u,w)\right]$$



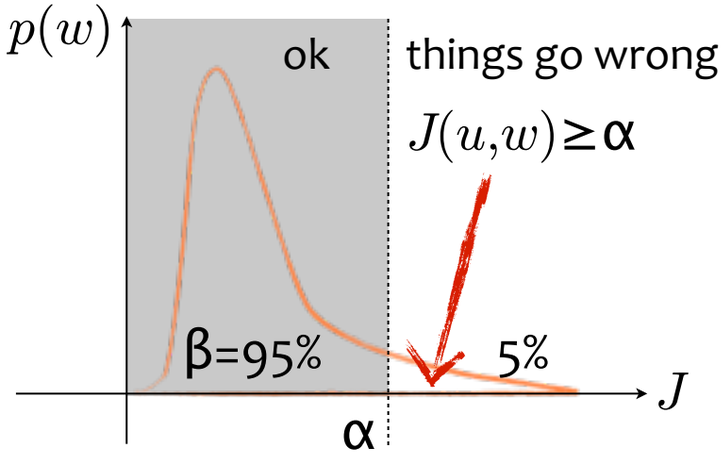
- Tradeoff between **expected performance & risk**

$$\min_u E_w\left[J(u,w)\right] + \rho\,\mathrm{Var}\left[J(u,w)\right]$$

- Conditional Value-at-Risk (**CVaR**)

$$\min_{u,\alpha}\left\{\alpha + \frac{1}{1-\beta}E[\max(J(u,w)-\alpha,0)]\right\}$$ (Rockafellar, Uryasev, 2000)

= minimize expected loss when things go wrong (convex if J convex !)



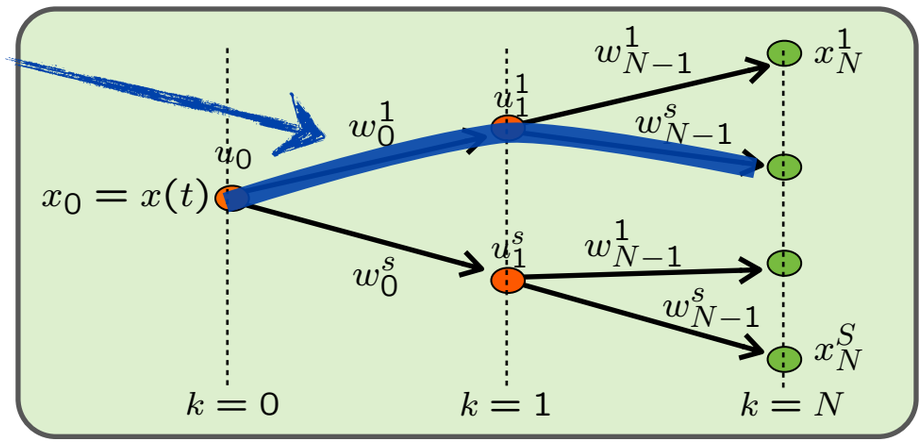
- **Min-max**

$$\min_u\{\max_w J(u,w)\}$$ =minimize worst case performance

Stochastic optimal control problem

- Enumerate all possible scenarios $\{w_0^j, w_1^j, \ldots, w_{N-1}^j\},\ j = 1, \ldots, S$

scenario #j



- Each scenario has probability

$$p^j = \prod_{k=0}^{N-1} \mathbf{Pr}[w_k = w_k^j]$$

- Each scenario has its own evolution $x_{k+1}^j = A(w_k^j)x_k^j + B(w_k^j)u_k^j$

(=linear time-varying system)

- Expectations become simple sums !

Ex: $\min E_w \left[x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \right]$

$$\min \sum_{j=1}^{S} p^j \left((x_N^j)' P x_N^j + \sum_{k=0}^{N-1} (x_k^j)' Q x_k^j + (u_k^j)' R u_k^j \right)$$

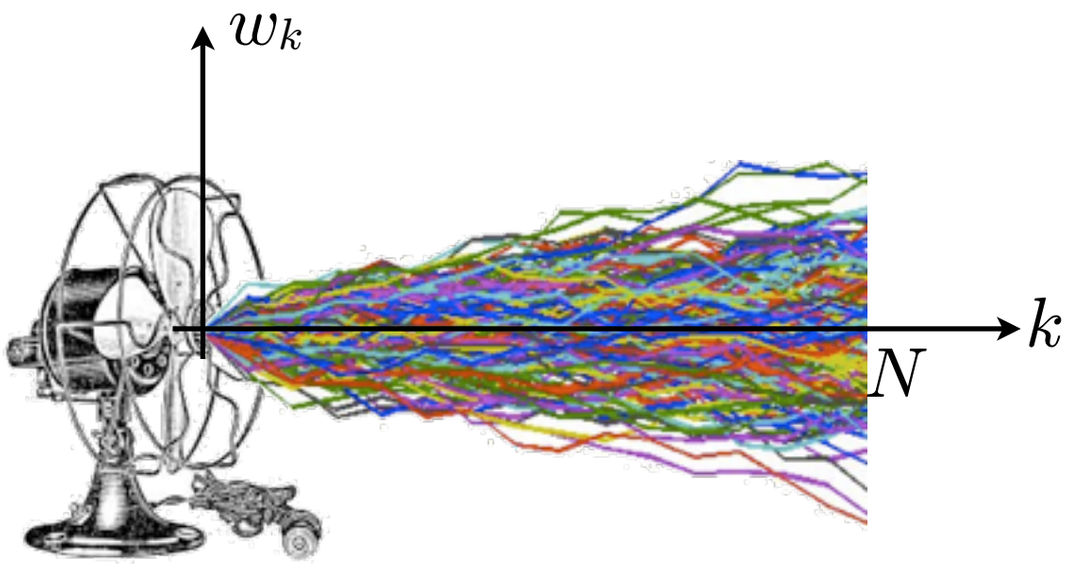
Expectations of quadratic costs remain quadratic costs

Scenario tree generation from data

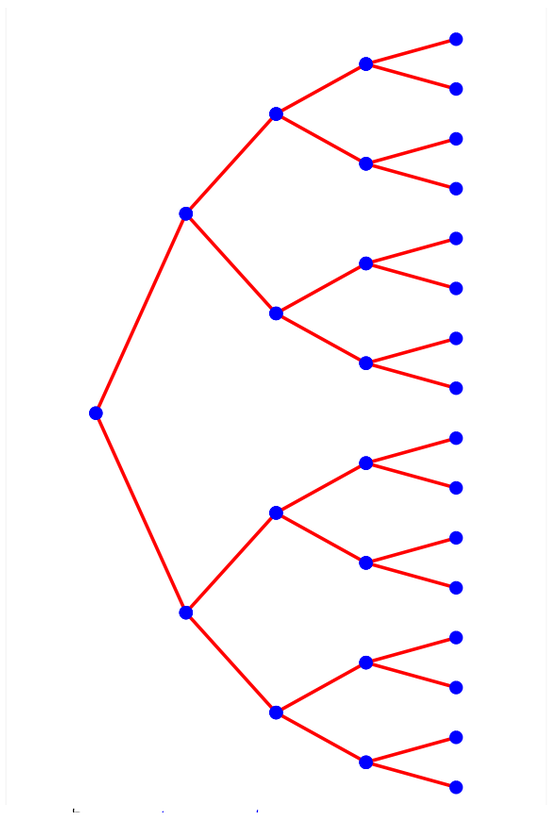
- Scenario trees can be generated by **clustering** sample paths (Heitsch, Römisch, 2009)

- Paths can be obtained by **Monte Carlo simulation** of (estimated) models, or from **historical data**

- The **number of nodes** can be decided a priori

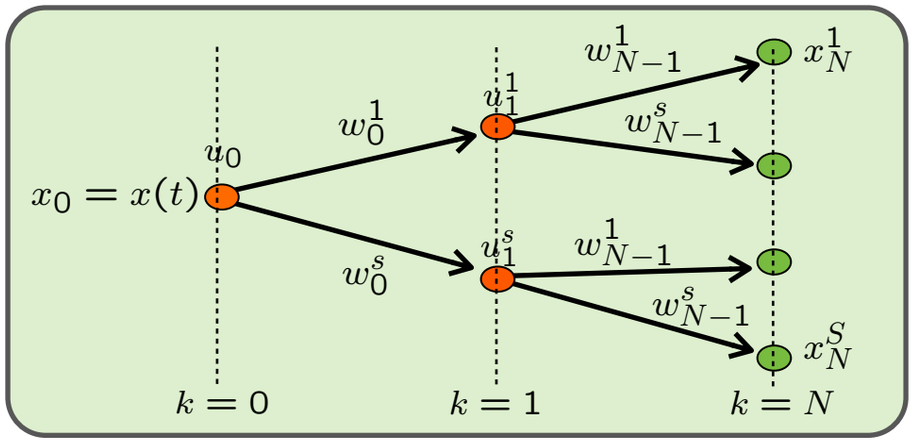


scenario "fan" (collection of sample paths)



scenario tree

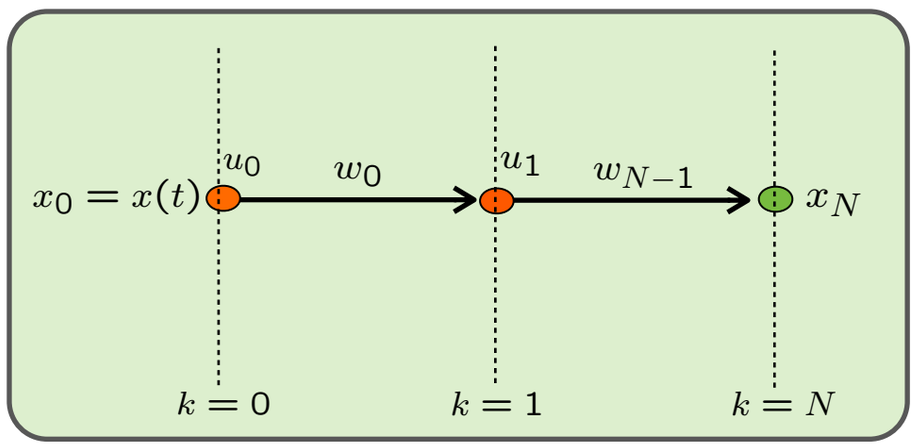
Free control variables



Stochastic optimal control

Causality constraint: $u_k^j = u_k^h$ when scenarios j and h share the same node at prediction time k (for example: $u_0^j \equiv u_0^h$ at root node $k{=}0$)

Decision u_k only depends on past disturbance realizations $\{w_0, w_1, \ldots, w_{k-1}\}$



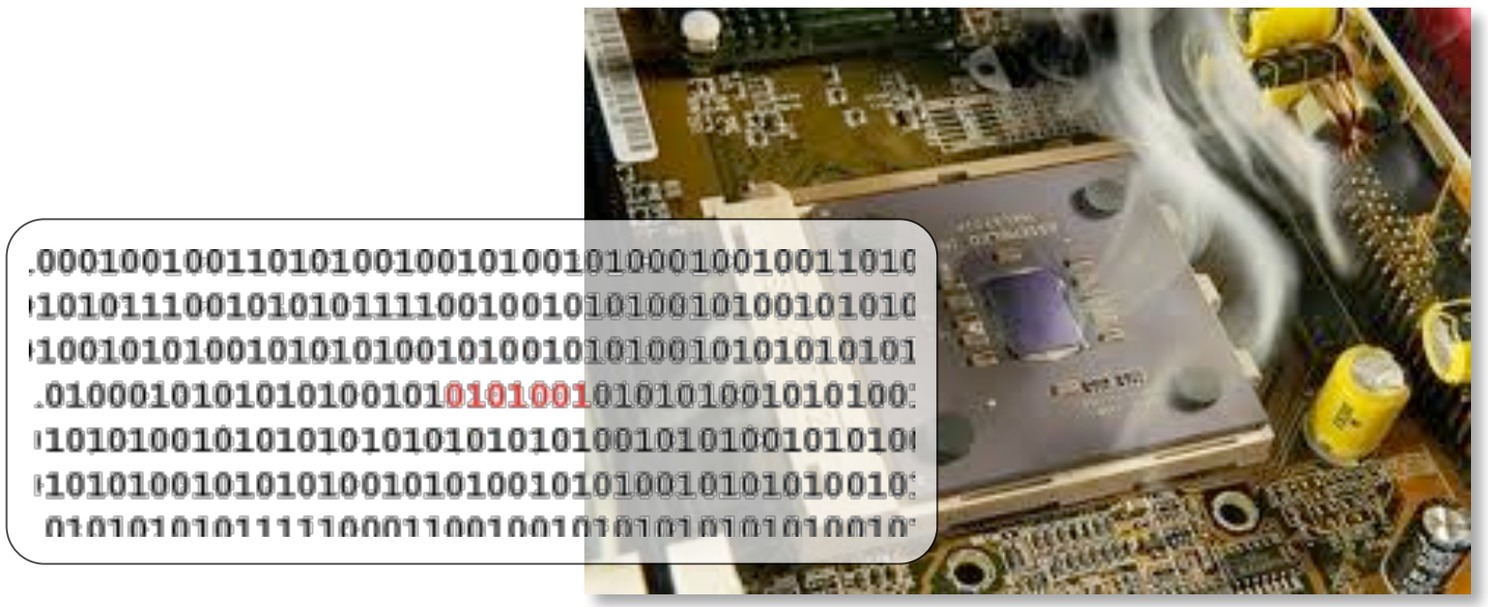
Deterministic control

Only a sequence of disturbances is considered

- frozen-time: $w_k \equiv w(t),\ \forall k$ (causal prediction)
- prescient control: $w_k \equiv w(t+k)$ (non-causal)
- certainty equivalence: $w_k = E[w(t+k)|t]$ (causal)

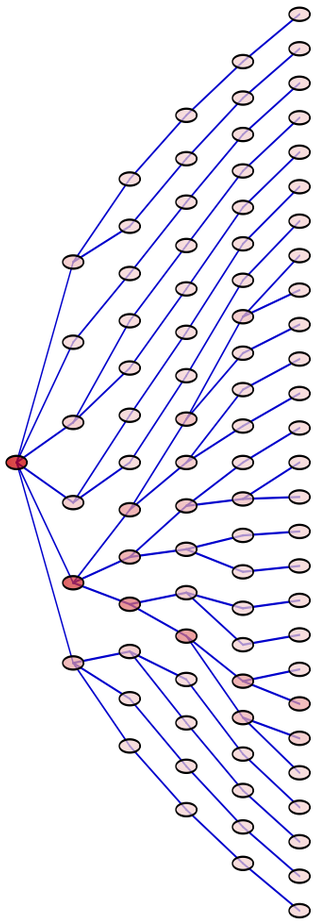
We can trade off between **complexity** of optimization problem (=number of nodes) and **performance** (=accuracy of stochastic modeling)

How about computation complexity ?



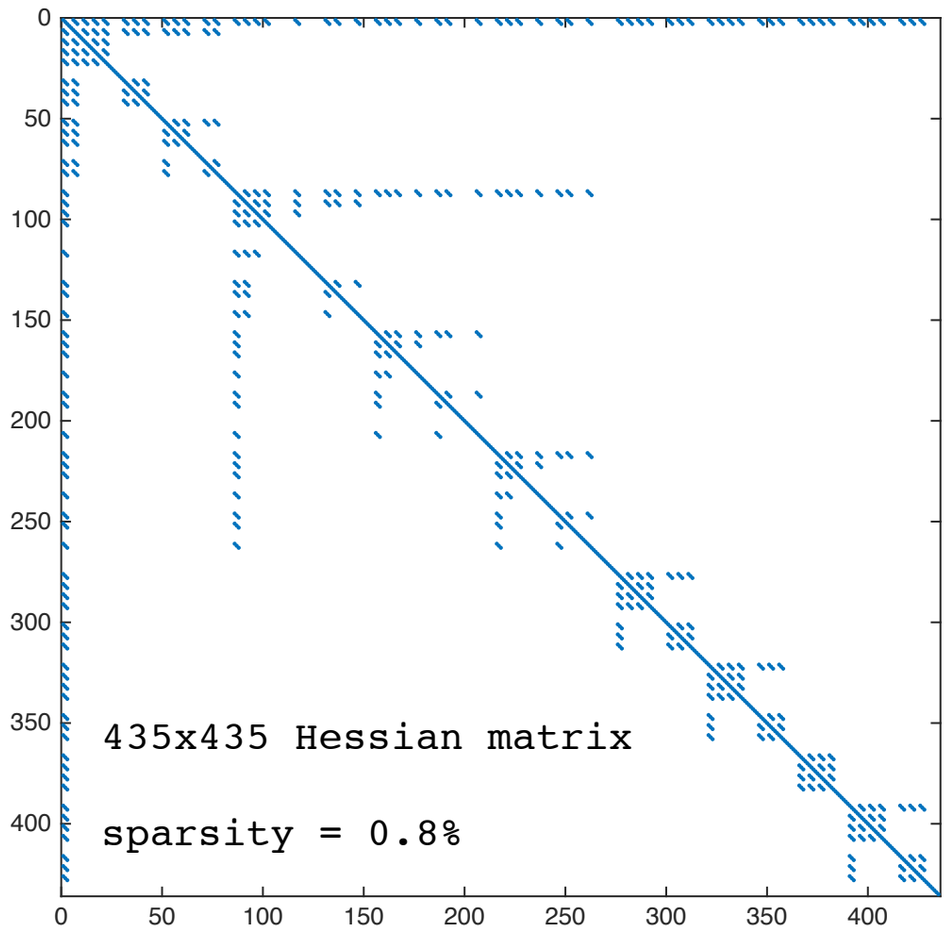
Complexity of optimization problem

- #optimization variables = #nodes x #inputs (in condensed version)
- Problems are **very sparse** (well exploited by **interior point methods**)
- Example: SMPC with quadratic cost and linear constraints

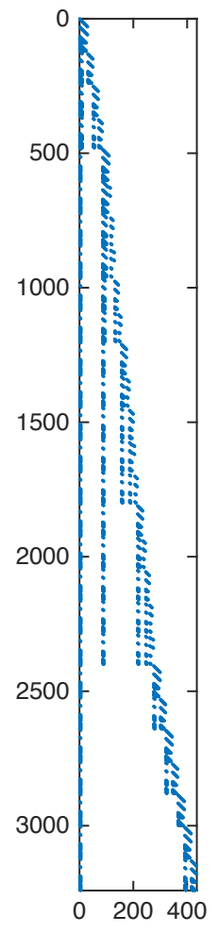


Tree=87 nodes

Branching factor M=[6 3 2 2 2]



435 free variables (5 inputs x node)



3240x435 constraint matrix
sparsity = 1.1%

Accelerated gradient projection algorithm for QP

(Patrinos, Bemporad, IEEE TAC, 2014)

- The SMPC problem can be compactly written as

$$\mathbb{P}(p): \; V^\star(p) = \min_{z \in \mathcal{Z}(p)} V(z) \quad \text{s.t.} \quad g(z) \le 0$$

constraints to be dualized

dynamics and non-anticipativity constraints

p = current state

$$x_{k+1} = A(w_k)x_k + B(w_k)u_k + f(w_k)$$

- *Lagrangean* function: $\mathcal{L}(z,y) = V(z) + y'g(z)$

dual function: $\Psi(p,y) = \min_{z \in \mathcal{Z}(p)} \mathcal{L}(z,y)$

- Dual problem: $\mathbb{D}(p): \Psi^\star(p) = \max_{y \ge 0} \Psi(p,y)$

By strong duality: $V^\star(p) = \Psi^\star(p)$

Accelerated gradient projection algorithm

(Patrinos, Bemporad, IEEE TAC, 2014)

- **GPAD** = **A**ccelerated **G**radient **P**rojection applied to **D**ual concave maximization problem

- Main properties of GPAD algorithm:
  - Fast global convergence
  - Easy to trade off accuracy for speed
  - Only *matrix-vector* operations required online
  - Complexity certification (tight upper bounds on worst-case #iterations)
  - Numerically stable (the non-accelerated version even works in fixed-point)
  - **Parallelizable** !

Distributed GPAD algorithm for Stochastic QP

0. $y_0 \leftarrow 0,\ \nu \leftarrow 0$

1. while $\max g(z_\nu) > \epsilon_g$ or $-\omega_\nu' g(z_\nu) > \epsilon_V$ do

   1.1. $\omega_\nu \leftarrow y_\nu + \theta_\nu(\theta_{\nu-1}^{-1} - 1)(y_\nu - y_{\nu-1})$ parallelizable !

   1.2. $z_\nu \leftarrow \arg\min_{z \in \mathcal{Z}(p)} \mathcal{L}(z, \omega_\nu)$ parallelizable !

   1.3. $y_\nu \leftarrow \max\{\omega_\nu + \frac{1}{L_\psi} g(z_\nu), 0\}$ parallelizable !

   1.4. $\theta_{\nu+1} \leftarrow \frac{1}{2}\left(\sqrt{\theta_\nu^4 + 4\theta_\nu^2} - \theta_\nu^2\right)$

2. end while

equality-constrained QP (solved by DP)

L_ψ is a Lipschitz constant of the dual gradient function

(various methods to compute L_ψ exist)

dim y = total # of *inequality constraints*

dim z = # nodes x (# states + # inputs)

Complexity:

The dual gradient is computed with complexity:

- Linear in the prediction horizon
- Linear in the size of the tree & is parallelizable
- Quadratic in the number of states and inputs

Memory requirements:

A few GB of offline data for problems with tens of state/input variables, hundreds of constraints and thousands of scenarios.

Distributed dual gradient projection

An ***offline*** factorization step is executed ***once*** to generate the matrices necessary for the ***fast online solution*** of the SMPC problem

Algorithm 7 Offline factorization for dGPAD

Require: A, B, N, F, G, F_N, P, tree structure for $w_k^{(i)}$

% For stage $N-1$ do

$\tilde{R}_{N-1} \leftarrow R + B'PB$

$L_{N-1} \leftarrow \text{cholesky}(\tilde{R}_{N-1})$

Solve $L_{N-1}L'_{N-1}K_{N-1} = -B'PA$ for K_{N-1}

$d_{N-1}^{(i)} \leftarrow F + GK_{N-1}$

$M_{N-1} \leftarrow A + BK_{N-1}$

$f_{N-1} \leftarrow F_N M_{N-1}$

$\Gamma_{N-1} \leftarrow K'_{N-1}RK_{N-1}$

$P_{N-1} \leftarrow Q + K_{N-1}^{(i)\prime}RK_{N-1} + M'_{N-1}PM_{N-1}$

for $i = 1, \dots, \mu_{N-1}$ **do**

% do in parallel for all i

$r \leftarrow \sum_{l \in \text{child}(i,N-1)} p_N^{(l)} w_{N-1}^{(l)}$

$c_{N-1}^{(i)} \leftarrow 2fM'_{N-1}Pr$

$h_{N-1}^{(i)} \leftarrow 2p_{N-1}^{(i)}A'PB + \Gamma_{N-1}$

$P_{N-1}^{(i)} \leftarrow p_{N-1}^{(i)}P_{N-1}$

Solve $L_{N-1}L'_{N-1}\Phi_{N-1}^{(i)} = -(2p_{N-1}^{(i)})^{-1}G'$ for $\Phi_{N-1}^{(i)}$

Solve $L_{N-1}L'_{N-1}\Theta_{N-1}^{(i)} = -(2p_{N-1}^{(i)})^{-1}B'F'_N$ for $\Theta_{N-1}^{(i)}$

Solve $L_{N-1}L'_{N-1}\sigma_{N-1}^{(i)} = -(2p_{N-1}^{(i)})^{-1}B'Pr$ for $\sigma_{N-1}^{(i)}$

end for

% Proceed backwards for all stages

for $k = N-1, \dots, 1$ **do**

$\tilde{R}_{k-1} \leftarrow R + B'P_kB$

$L_{k-1} \leftarrow \text{cholesky}(\tilde{R}_{k-1})$

Solve $L_{k-1}L'_{k-1}K_{k-1} = -B'P_kA$ for K_{k-1}

$f_{k-1} \leftarrow A + BK_{k-1}$

$P_{k-1} \leftarrow Q + K'_{k-1}RK_{k-1} + f'_{k-1}P_kf_{k-1}$

$d_k^{(i)} \leftarrow F + GK_{k-1}^{(i)}$

for $i = 1, \mu_k$ **do**

% Execute in parallel across all i

$r \leftarrow \sum_{l \in \text{child}(i,k-1)} p_k^{(l)} w_{k-1}^{(l)}$

$c_{k-1}^{(i)} \leftarrow 2f'_{k-1}P_kr$

$h_{k-1}^{(i)} \leftarrow 2p_{k-1}^{(i)}B'P_kA + R_{k-1}^{(i)}K_{k-1}^{(i)}$

Solve $L_{k-1}L'_{k-1}\Phi_{k-1}^{(i)} = -(2p_k^{(i)})^{-1}G'$ for $\Phi_{k-1}^{(i)}$

Solve $L_{k-1}L'_{k-1}\Theta_{k-1}^{(i)} = -(2p_{k-1}^{(i)})^{-1}B'$ for $\Theta_{k-1}^{(i)}$

Solve $L_{k-1}L'_{k-1}\sigma_{k-1}^{(i)} = -(2p_{k-1}^{(i)})^{-1}B'Pr$ for $\sigma_{k-1}^{(i)}$

end for

end for

return $\Phi_k^{(i)}$, $\Theta_k^{(i)}$, $h_k^{(i)}$, $\sigma_k^{(i)}$, K_k for all k and i

offline

Algorithm 8 Online computation of the dual gradient at the current primal-dual iterate of the dGPAD algorithm - distributed implementation

Require: p (initial state), N, $\Phi_k^{(i)}$, $\Theta_k^{(i)}$, $c_k^{(i)}$, $h_k^{(i)}$, $\sigma_k^{(i)}$, K_k, y_k^i for all k and i, tree structure for $w_k^{(i)}$, A, B

*% **Backward iteration***

% Stream data from the host (CPU) to the device (GPU) for:

% $\Phi_k^{(i)}$, $\Theta_k^{(i)}$, $\sigma_k^{(i)}$, $c_k^{(i)}$, $d_k^{(i)}$, $f_k^{(i)}$ and $h_k^{(i)}$.

for $k = N-1, \dots, 0$ **do**

for $i = 1, \dots, \mu_k$ **do**

% In parallel for all i

$s_k^{(i)} \leftarrow \Phi_k^{(i)}y_k^{(i)} + \Theta_k^{(i)} \sum_{l \in \text{child}(i,k)} q_{k+1}^{(l)} + \sigma_k^{(i)}$ *% where $q_N^{(i)} \equiv y_N^{(i)}$*

$q_k^{(i)} \leftarrow c_k^{(i)} + d_k^{(i)\prime}y_k^{(i)} + f_k^{(i)\prime} \sum_{l \in \text{child}(i,k)} q_{k+1}^{(l)} + h_k^{(i)\prime}s_k^{(i)}$

end for

end for

Backward Iteration for the dual variables

% Free memory allocated for:

% $\Phi_k^{(i)}$, $\Theta_k^{(i)}$, $\sigma_k^{(i)}$, $c_k^{(i)}$, $d_k^{(i)}$, $f_k^{(i)}$ and $h_k^{(i)}$

*% **Forward iteration** (x_0 is given)*

% Load to the device data for K_k, s_k, A, B, and $w_k^{(l)}$.

$x_0 \leftarrow p$

for $k = 0, \dots, N-1$ **do**

for $i = 1, \dots, \mu_k$ **do**

% In parallel for all i

$u_k^{(i)\star} \leftarrow K_k x_k^{(i)\star} + s_k^{(i)}$

for $l \in \text{child}(i,k)$ **do** {:L1}

$x_{k+1}^{(l)\star} \leftarrow Ax_k^{(i)\star} + Bu_k^{(i)\star} + w_k^{(l)}$

end for

end for

end for

return $z^\star$, *i.e.*, the sequence of $x_k^{(i)\star}$ and $u_k^{(i)\star}$.

Forward iteration: system simulation

OnLine

only matvec operations

Distributed dual gradient projection

The algorithm yields an ε_g-infeasible, ε_V-suboptimal solution in the sense:

$$V(z) - V^\star \leq \epsilon_V, \qquad \text{optimality tolerance}$$

$$g_i(z) \leq \epsilon_g, \ \forall i \qquad \text{feasibility tolerance}$$

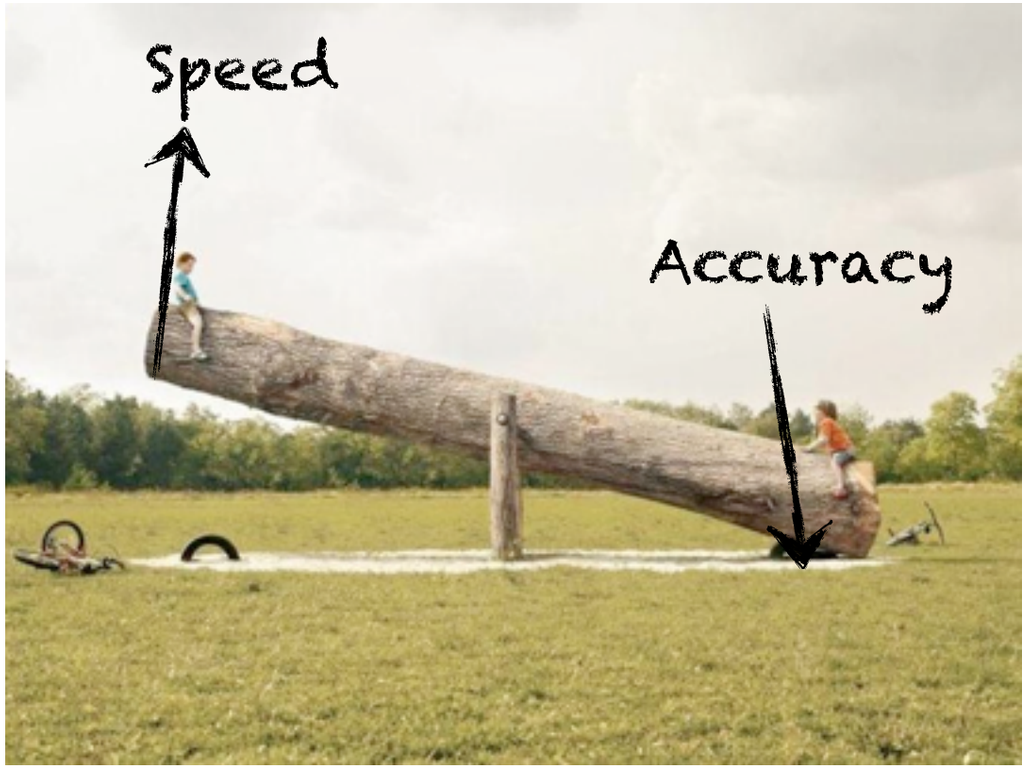
V^* is unknown, so we make use of the duality gap as a stopping criterion:

$$V(z_\nu) - \Psi(y_{\nu+1}) \leq \epsilon_V \max\{1, \Psi(y_{\nu+1})\}$$

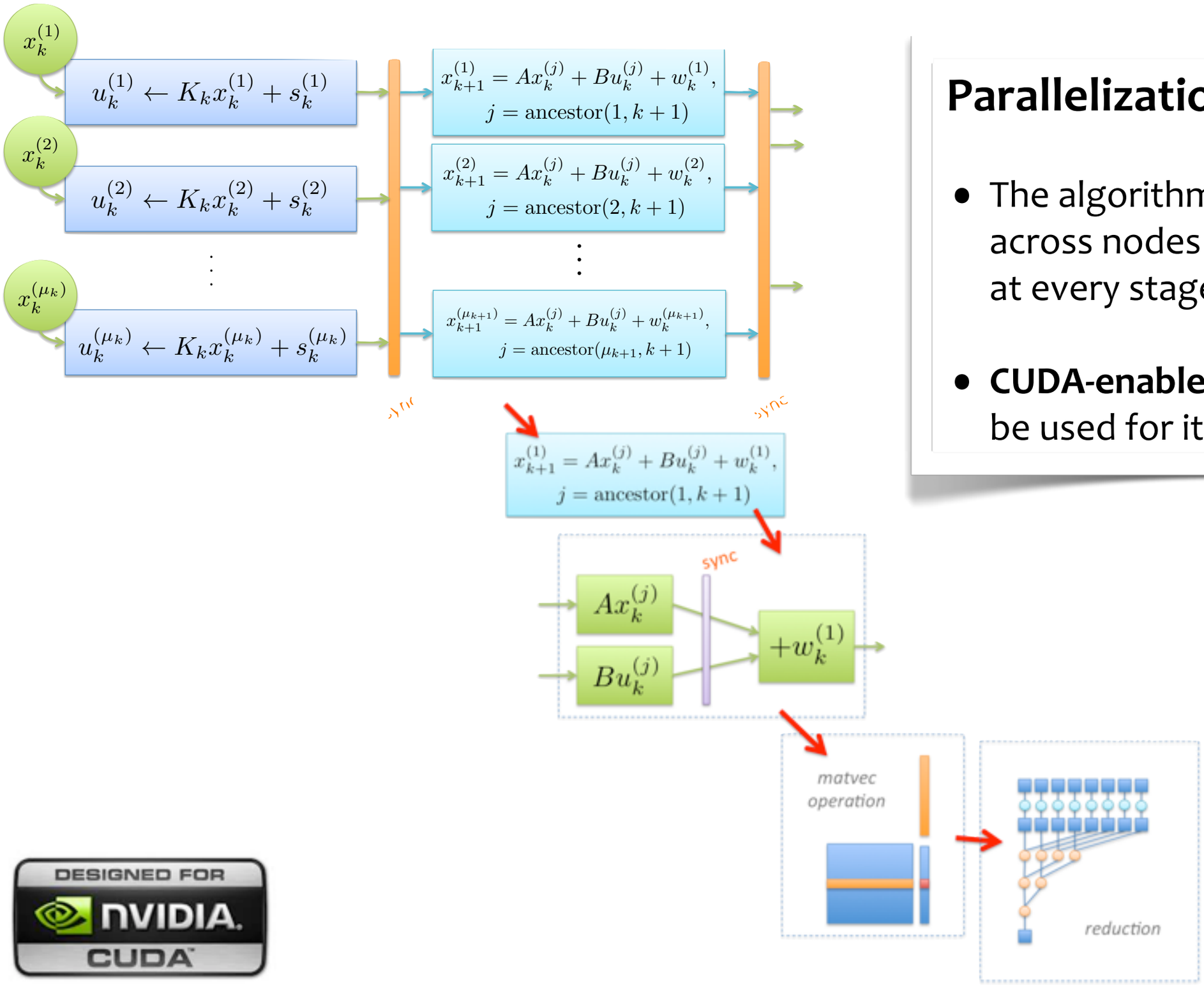
which is satisfied whenever:

$$-w_\nu' g(z_\nu) \leq \epsilon_V$$

Checking these conditions is computationally cheap



Distributed dual gradient projection

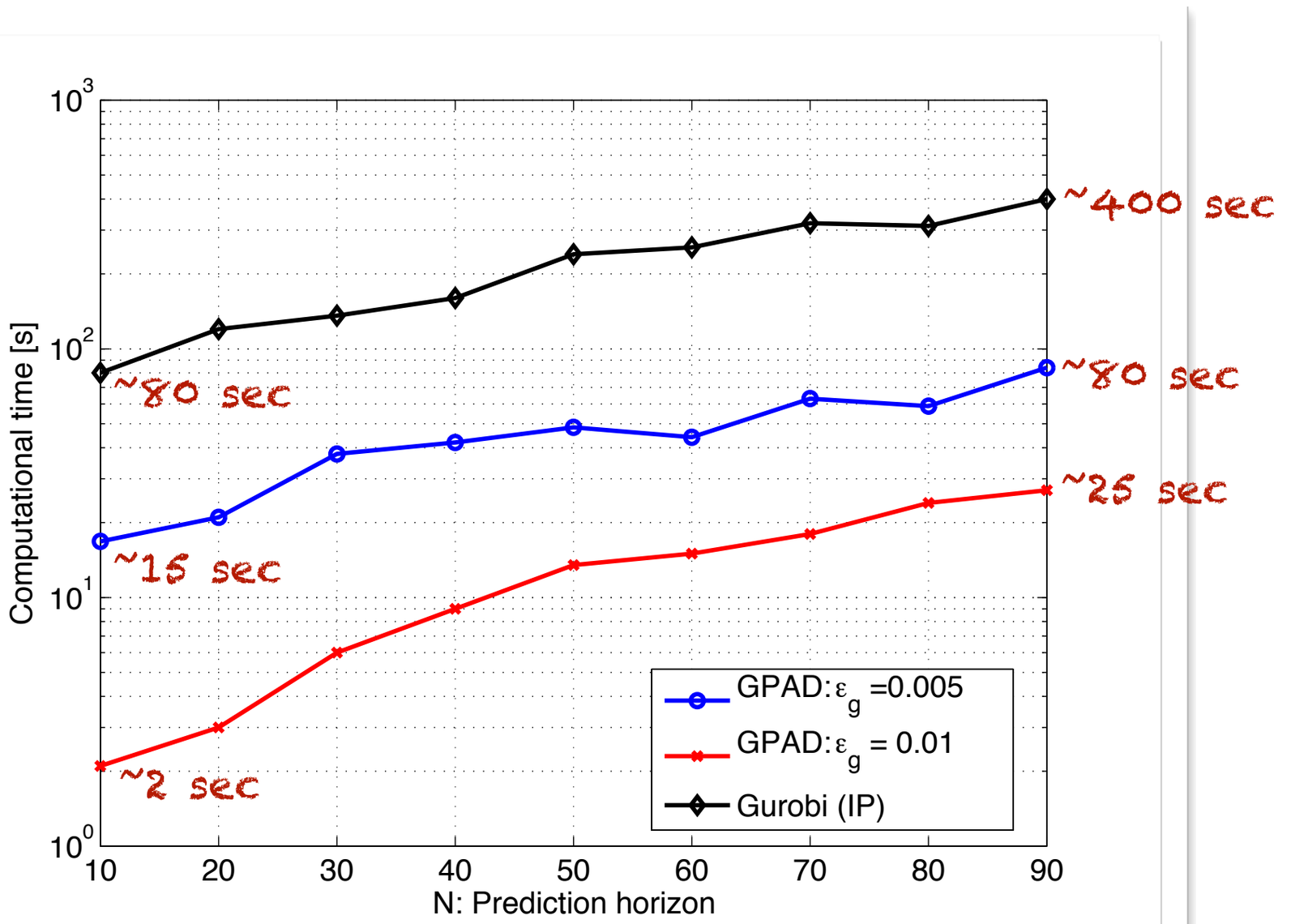


Parallelization:

- The algorithm can be parallelized across nodes of the scenario tree at every stage *k*
- **CUDA-enabled** GPD devices can be used for its implementation.

dGPAD: Evaluation of the algorithm

Problem size: linear stochastic system with **60 states, 25 inputs, 256 scenarios**

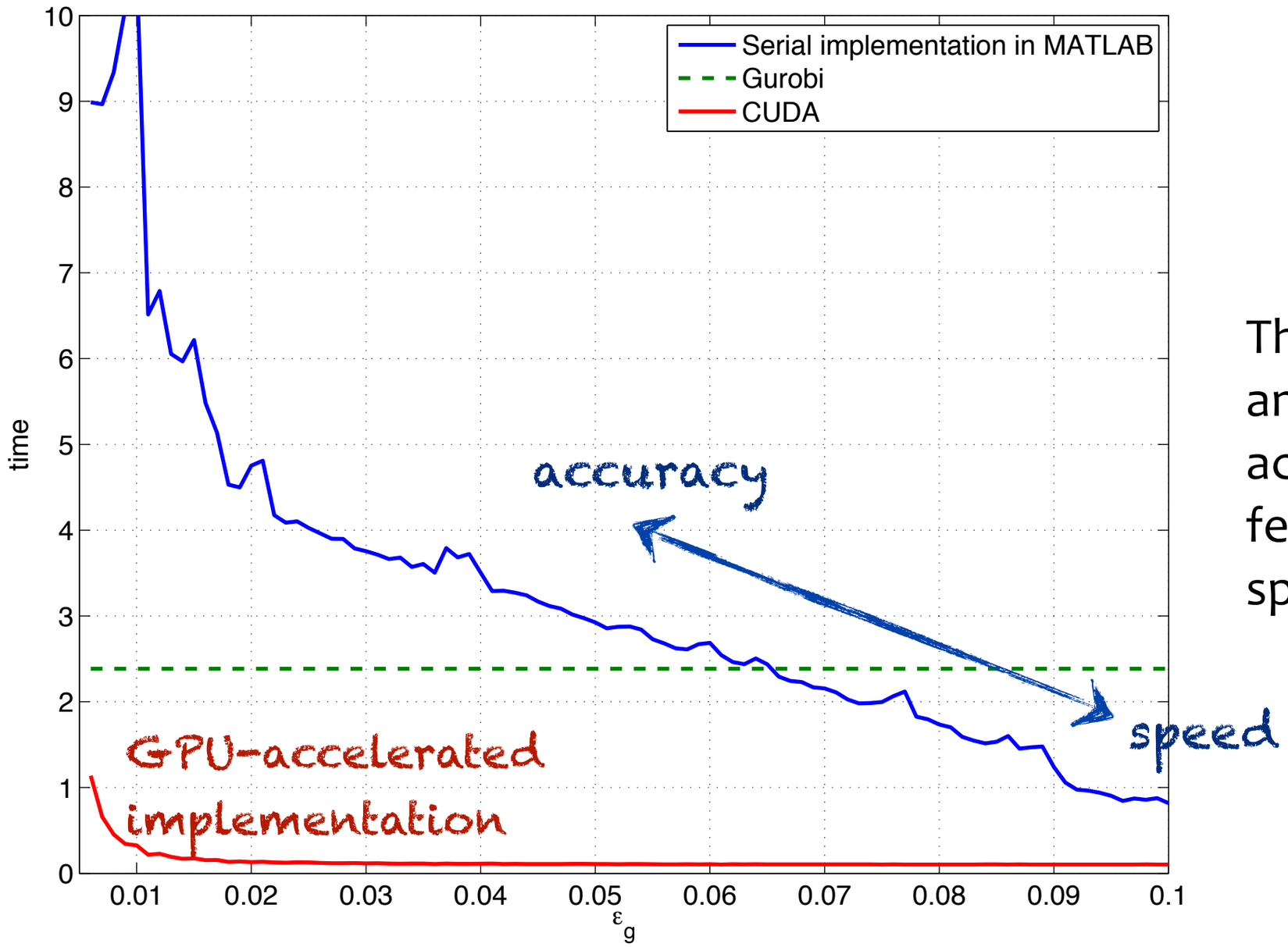


16x ÷ 40x faster
than commercial
state-of-the-art
Interior-Point method

Remark: For larger problems (*e.g.*, 50 states, 30 inputs, 9036 nodes)
GUROBI gets stuck on a 4GB 4-core PC, while dGPAD can solve the problem

dGPAD: Evaluation of the algorithm

Problem size: linear stochastic system with 30 states, 10 inputs, 72 scenarios (N=10)

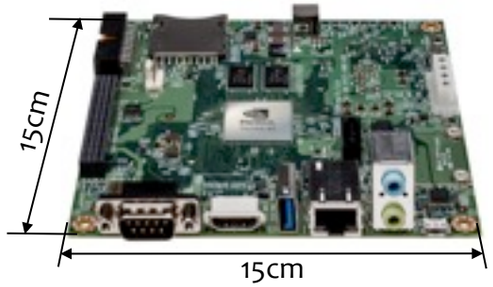


The tuning parameters ε_g and ε_V allow to trade off accuracy (optimality and feasibility) for solution speed

GPU computations

Tesla 2075
515 Gflops/s
1.15 Ghz, 6 GB DRAM
448 cores
~$2,300

SMPC can be solved by dGPAD on ***cheap*** **embedded** platforms to control complex large-scale systems (accounting for uncertainties in an optimal and risk-averse fashion)



Jetson TK1
192 CUDA cores
158 GFlops/s
852MHz
$192

Intel i5-4570
4 cores @ 3.2MHz
0.820 GFlops/s
~$550

Does it work in practice ?

A few sample applications of SMPC

- **Financial engineering**: dynamic hedging of portfolios replicating synthetic options

  (Bemporad, Bellucci, Gabbriellini, 2009)
  (Bemporad, Gabbriellini, Puglia, Bellucci, 2010)
  (Bemporad,Puglia, Gabbriellini, 2011)

- **Energy systems**: power dispatch in smart grids, optimal bidding on electricity markets

  (Patrinos, Trimboli, Bemporad 2011)
  (Puglia, Bernardini, Bemporad 2011)

- **Automotive control**: energy management in HEVs, adaptive cruise control (human-machine interaction)

  (Bichi, Ripaccioli, Di Cairano, Bernardini, Bemporad, Kolmanovsky, CDC 2010)

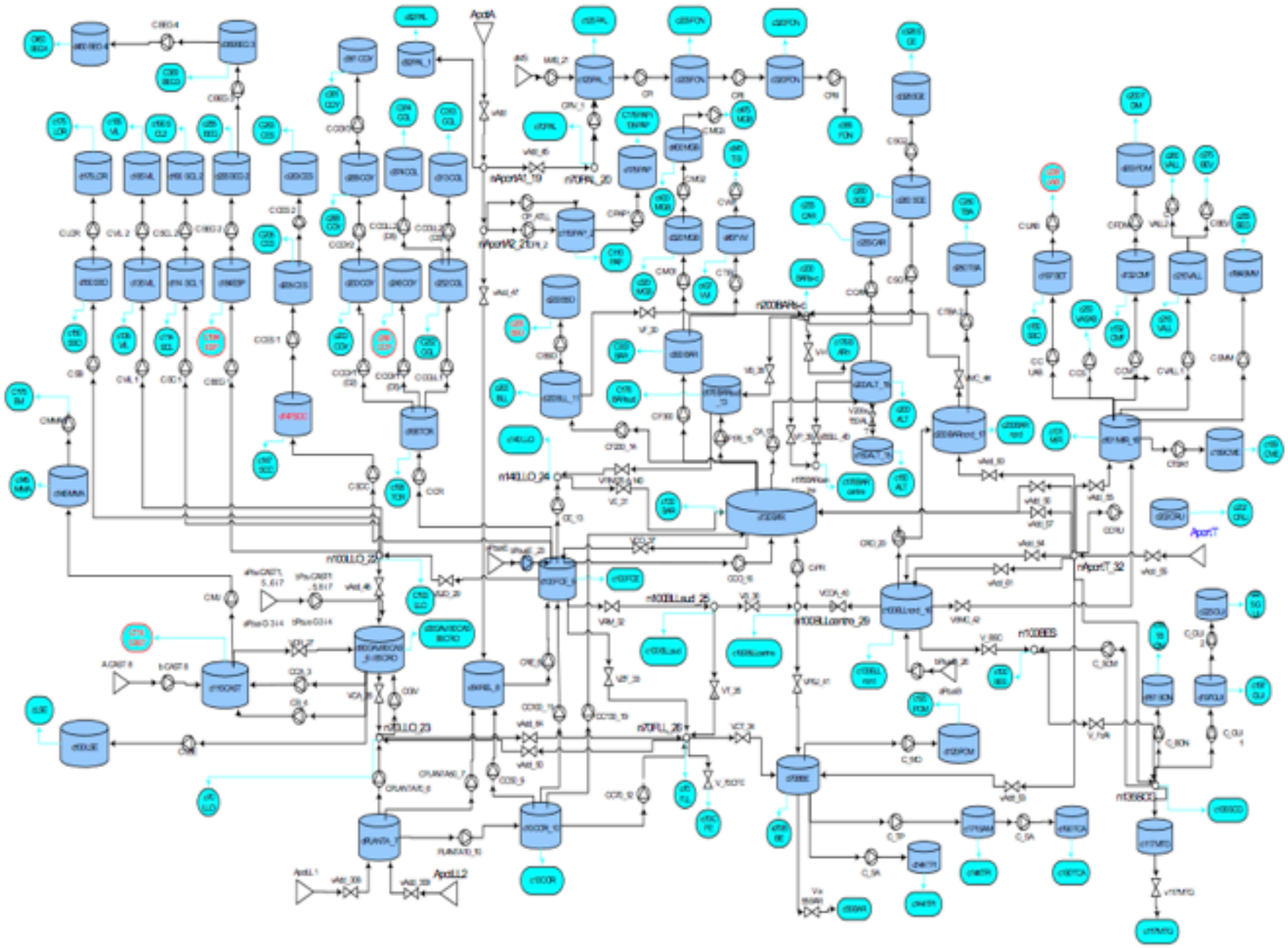
- **Networked control**: improve robustness against communication imperfections

  (Bernardini, Donkers, Bemporad, Heemels, NECSYS 2010)

- **Water networks**: pumping control in urban drinking water networks, under uncertain demand & minimizing costs under varying electricity prices

  (Sampathirao, Sopasakis, Bemporad, this talk)

Drinking water network of Barcelona (Spain)



- General overview:

Municipalities supplied	23
Supply area	424 km^2
Population supplied	2.922.773
Average demand	7 m^3/s

- Network parameters:

Pipes length	4.645 km
Pressure floors	113
Sectors	218

- Facilities

Remote stations	98
Water storage tanks	81
Valves	64
Flow meters	92
Pumps / Pumping stations	180 / 84
Chlorine dosing devices	23
Chlorine analyzers	74

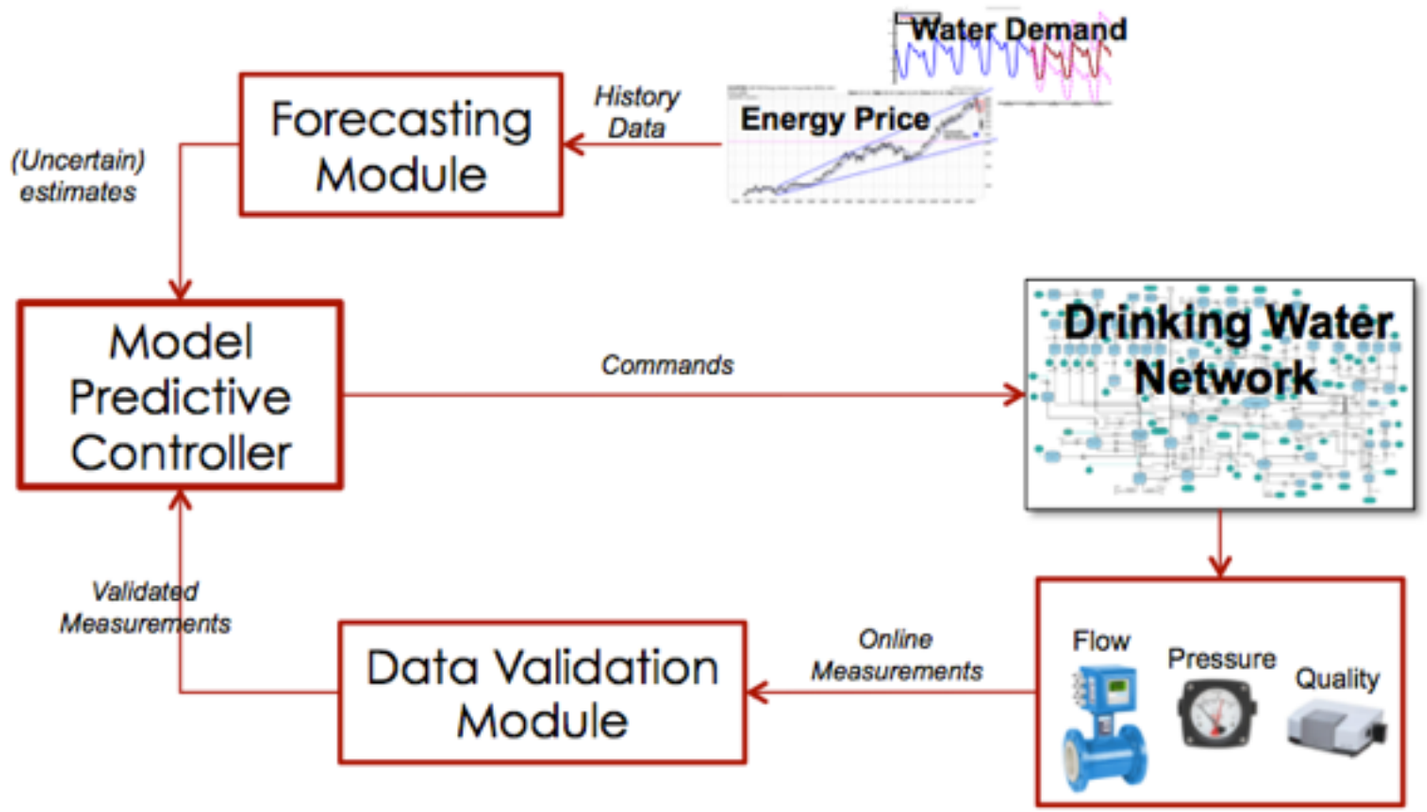
European FP7-ICT project WIDE
"DEcentralized and WIreless Control of Large-Scale Systems"

European FP7-ICT project EFFINET
"EFFIcient Integrated Real-time Monitoring and Control of Drinking Water NETworks"

Control of the drinking water network of BCN

Main Goals:

- Reduce **electricity consumption** for pumping (€ € €)
- Meet **demand** requirements
- Deliver **smooth** control actions
- Keep storage tanks above safety **limits**
- Respect the technical **limitations**: pressure limits, overflow limits & pumping capabilities



EFFINET

Control of the drinking water network of BCN

- The control objectives are translated into cost functions:

Expected total squared water production cost (ETSWPC) = **economic** cost

$$J^{ws} = W_\alpha^2 \sum_{l=1}^{K} \sum_{i=0}^{H_p-1} p^l (\alpha_1 + \alpha_{2,k})^2 (u^l_{k+i|k})^2$$

€

Expected total smooth **operation** cost (ETSOC)

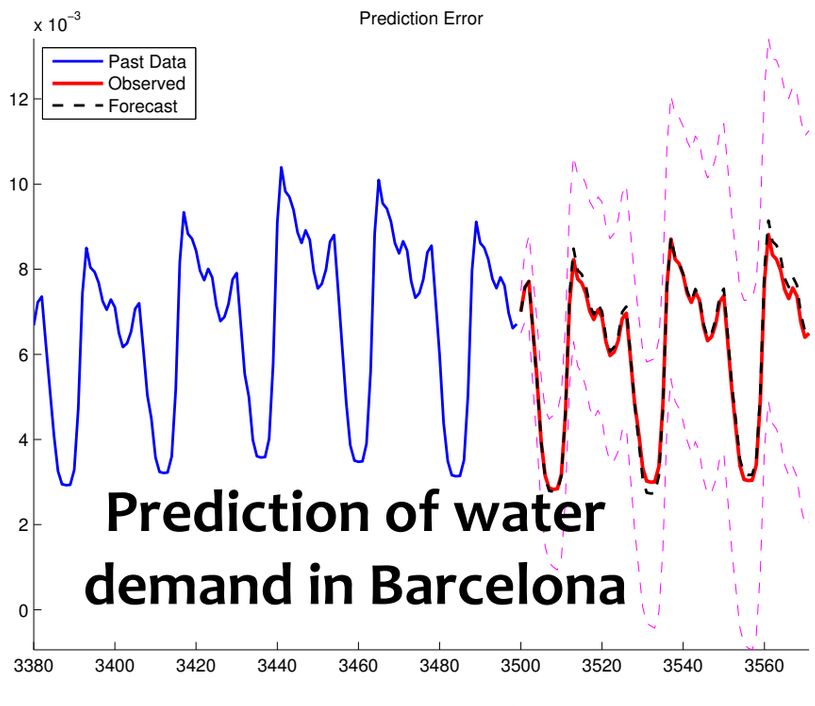
$$J^{\Delta} = \sum_{l=1}^{K} \sum_{i=1}^{H_p-1} p^l \ell^{\Delta}(\Delta u^l_{k+i|k})$$

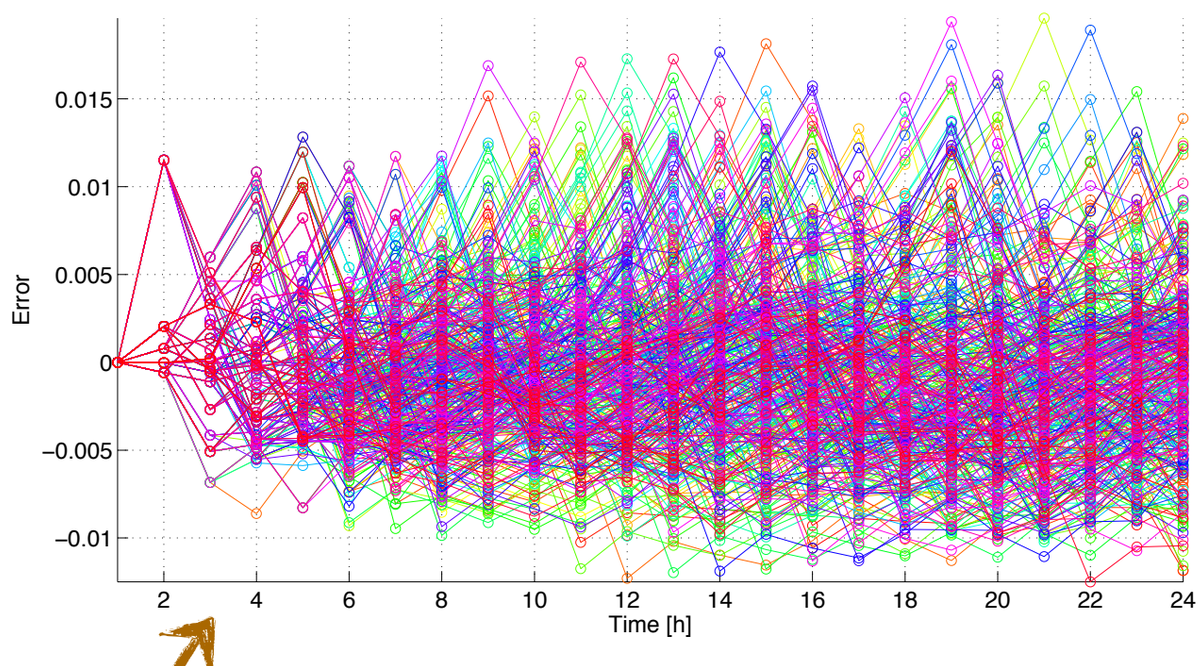
expected total **safety** storage cost (ETSSC)

$$J^{S} = \sum_{l=1}^{K} \sum_{i=1}^{H_p} p^l \ell^{s}(x^l_{k+i|k})$$

Need to minimize the total operating cost $V = J^{ws} + J^{\Delta} + J^{S}$

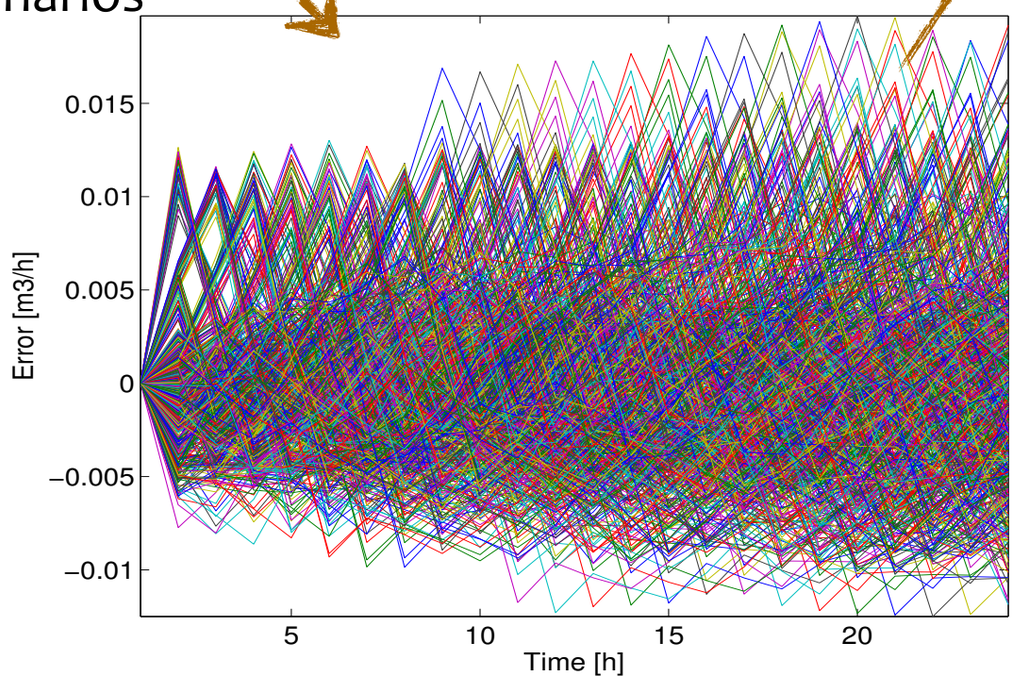
Control of the drinking water network of BCN





Uncertainty represented as a fan of scenarios

Reduction to a scenario tree

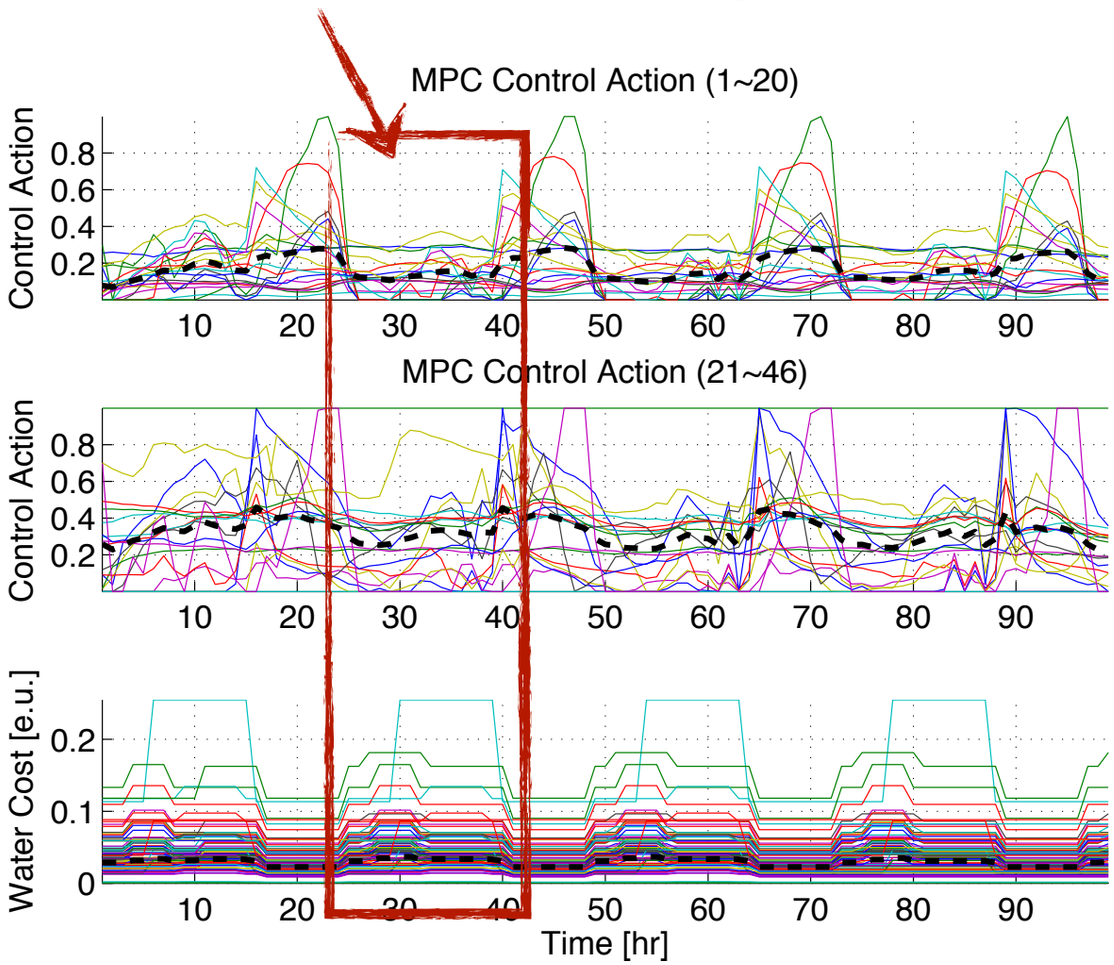


$$d_{k+i|k} = \hat{d}_{k+i|k} + \epsilon_{k+i|k}$$

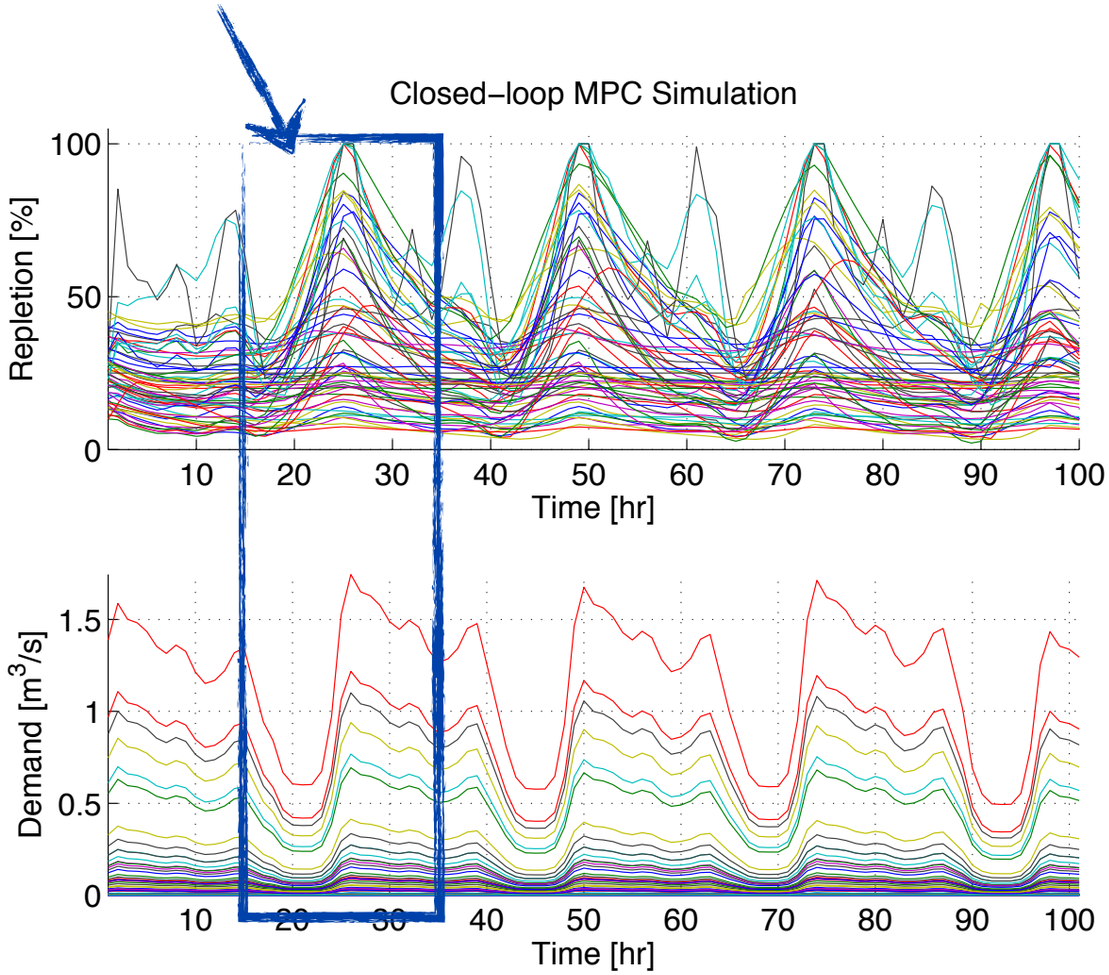
Uncertainty:
demand prediction error

Control of the drinking water network of BCN

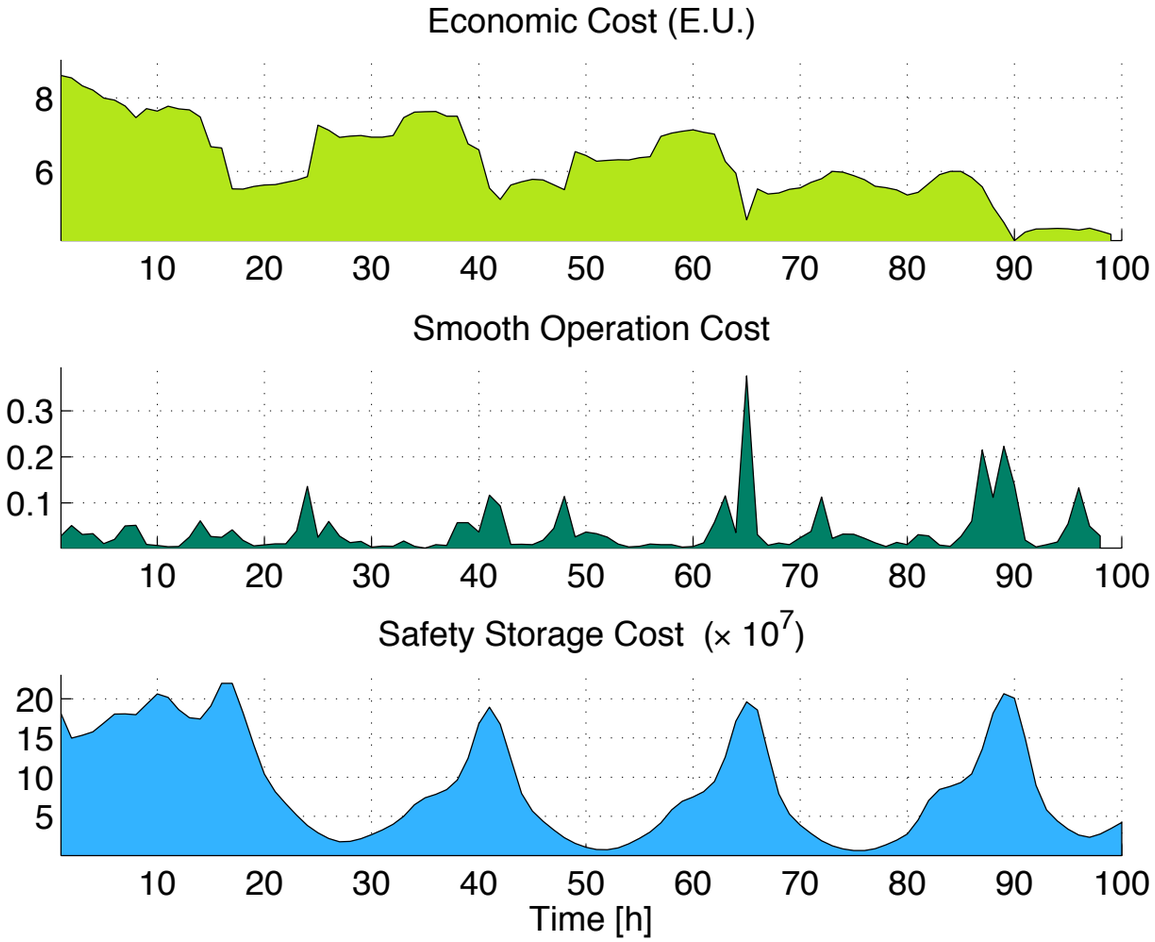
Economic: *Avoid pumping when the price of electricity is high*



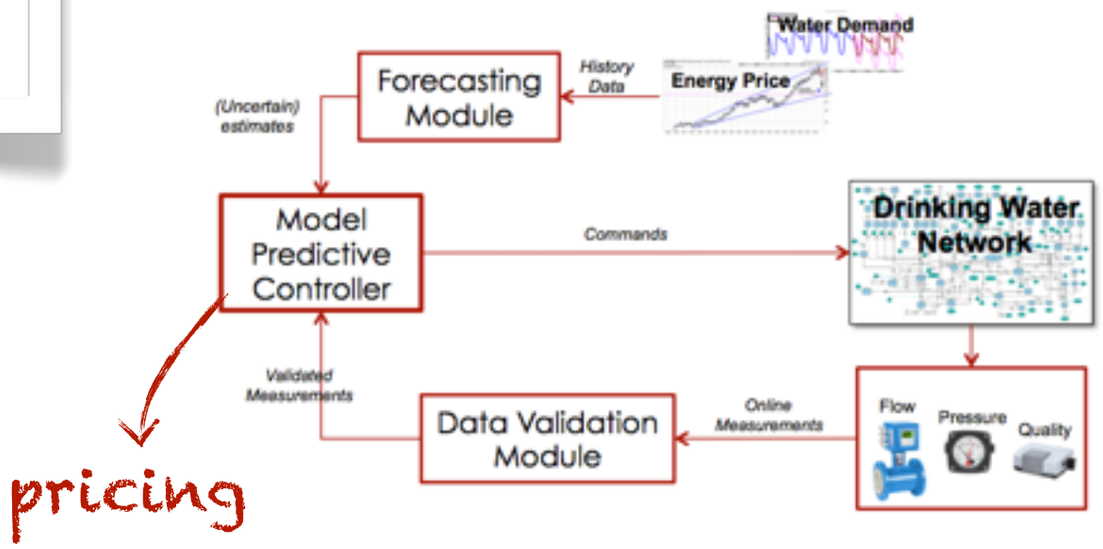
Foresight: *tanks start loading up before the consumers ask for water*



Control of the drinking water network of BCN



SMPC: The network operator has online information about the current and predicted operating cost in real time



Scalability of the SMPC approach

How does the approach **scale** with the dimension of the system ?

- The dGPAD algorithm scales-up well with the size of the scenario tree (thanks to heavy parallelization)

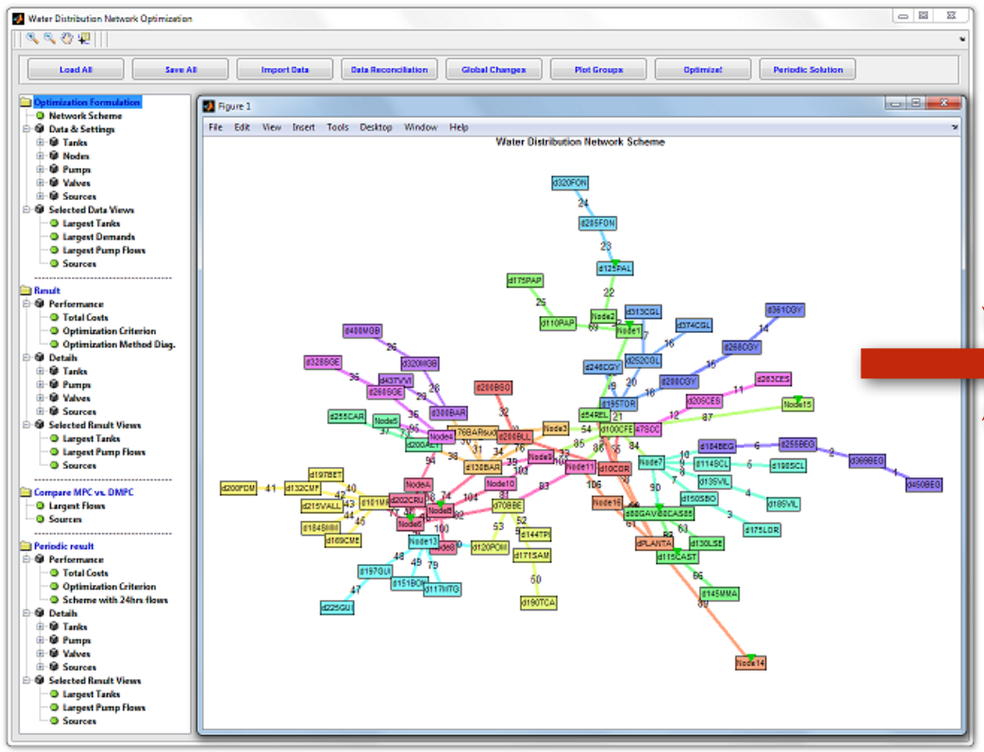
- Scalable alternatives:

  – **Decentralized SMPC**: divide into subsystems and control each of them in parallel, exchanging some decisions **after** computations (others' decisions = measured disturbances) (Bemporad, Barcelli, 2010)

  – **Distributed SMPC**: exchange some global variables **during** computations (Negenborn, Maestre, IEEE CSM, 2014)

- The same dGPAD algorithm can be used for decentralized SMPC (immediately), or for distributed SMPC by relaxing also the constraints that (weakly) couple the subsystems

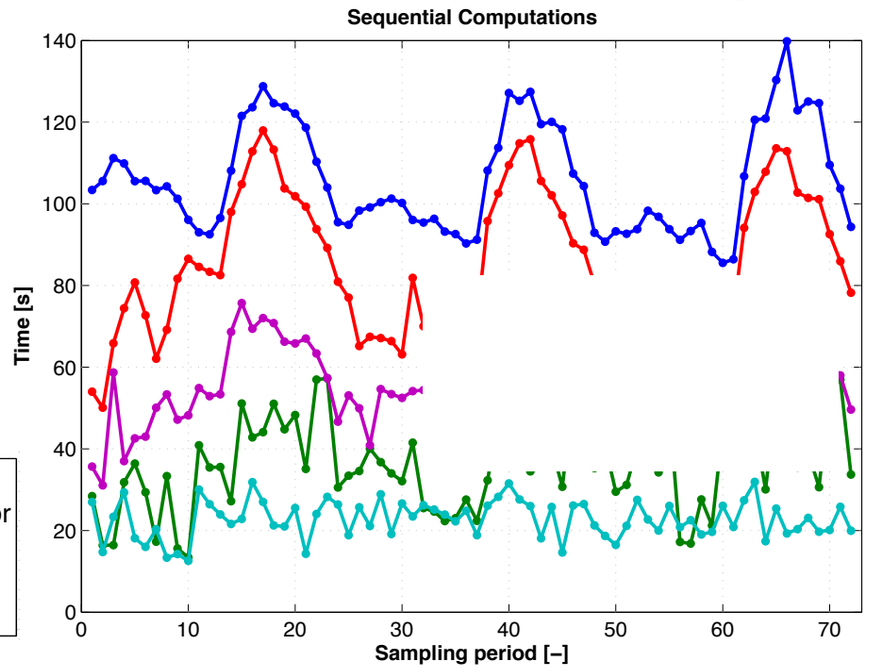
Distributed MPC of Barcelona Network



distributed MPC algorithm (optimal solution)

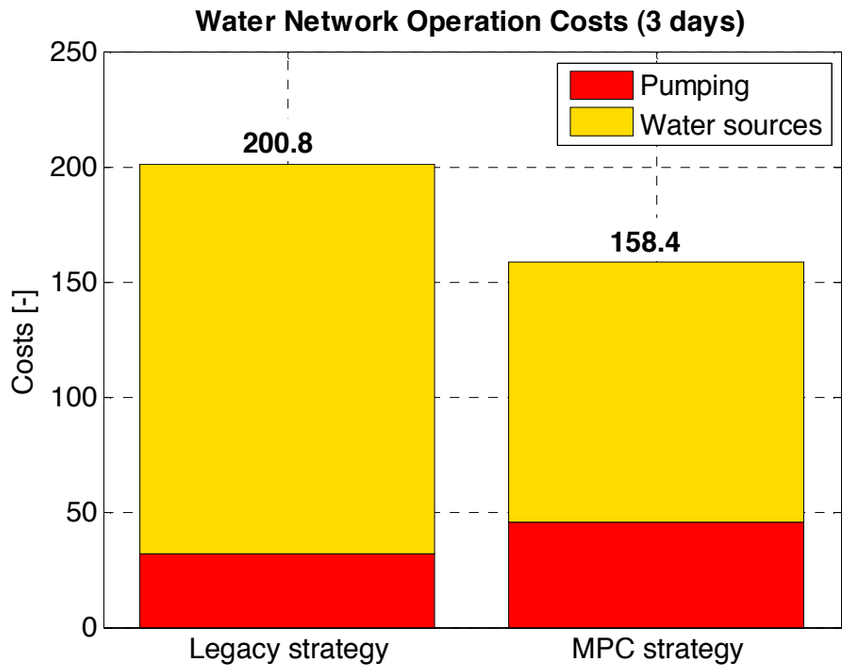
(Trnka, Pekar, Havlena, IFAC 2011)

Honeywell



Network separation to groups

- Benefits evaluated on 3 days historic data set
- Benefits evaluation is complicated by water accumulation (different final accumulation for different control strategies)
- To get comparable data MPC was forced to fill tanks to the same final levels as in historical data (sub-optimal)
- ~20% direct cost savings (pumping and water sources)
- Indirect savings by smooth MV's operation -> leakage prevention by small pressure surges and reduced equipment tear & wear



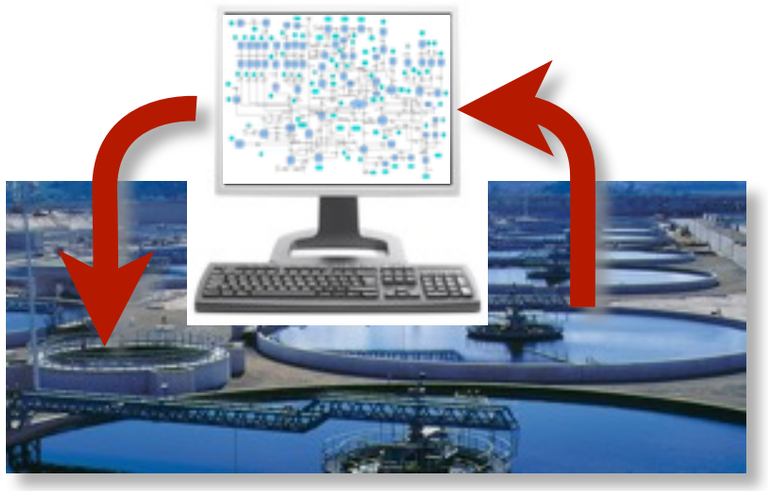
Conclusions

- SMPC is a very powerful control technique to deal with rather complex control problems (**large-scale, stochastic, constrained**) in an **optimal** way

- Optimal = **best decisions** based on **best predictions of uncertainty**

- Useful in many complex industrial problems (not only *process industries* but also *automotive*, *energy and smart grids*, *water nets*, …)

- Scalability ensured by massive **parallelization** of stochastic QP solvers



SMPC is a mature technology for complex control applications

- Generally speaking, several problems in complex networks can be dealt with by (huge-scale) convex optimization algorithms (Nesterov, 2012)