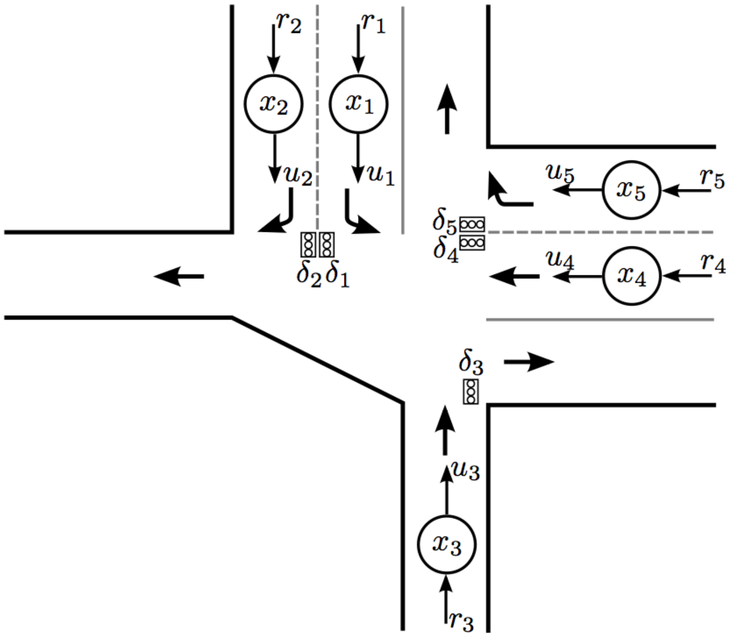
Embedded Optimization for Mixed Logic Dynamical Systems

Alexander Domahidi
Joint work with Damian Frick and Manfred Morari

EMBOPT Workshop
IMT Lucca, Italy
September 8, 2014

Application: Optimal Traffic Control

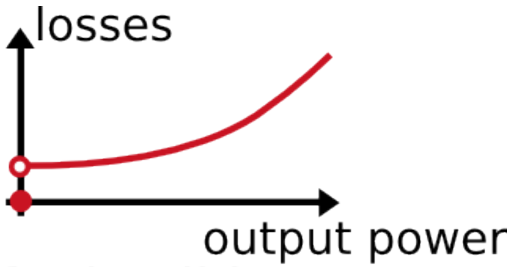


Source: sunshinekelly.com

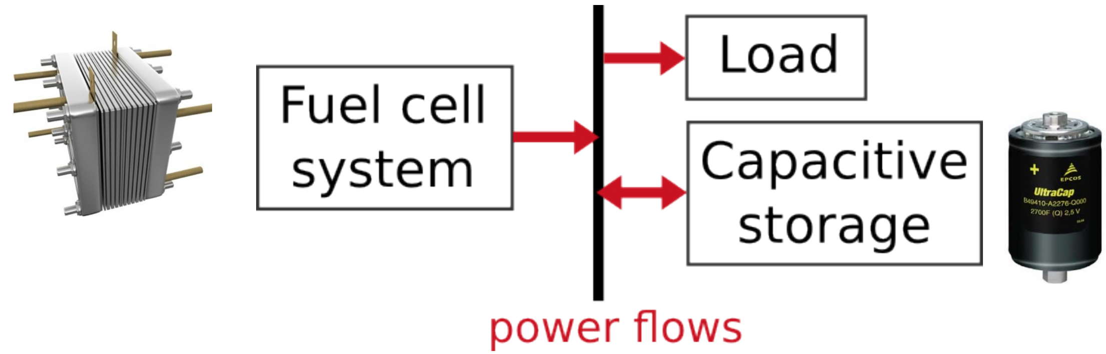
- Discrete signaling with switching cost
- Logical constraints on signaling
- **Control objective:** minimize queue size

Medium-to-large-scale hybrid MPC problem

Application: Fuel Cell Power Management



fuel cell losses



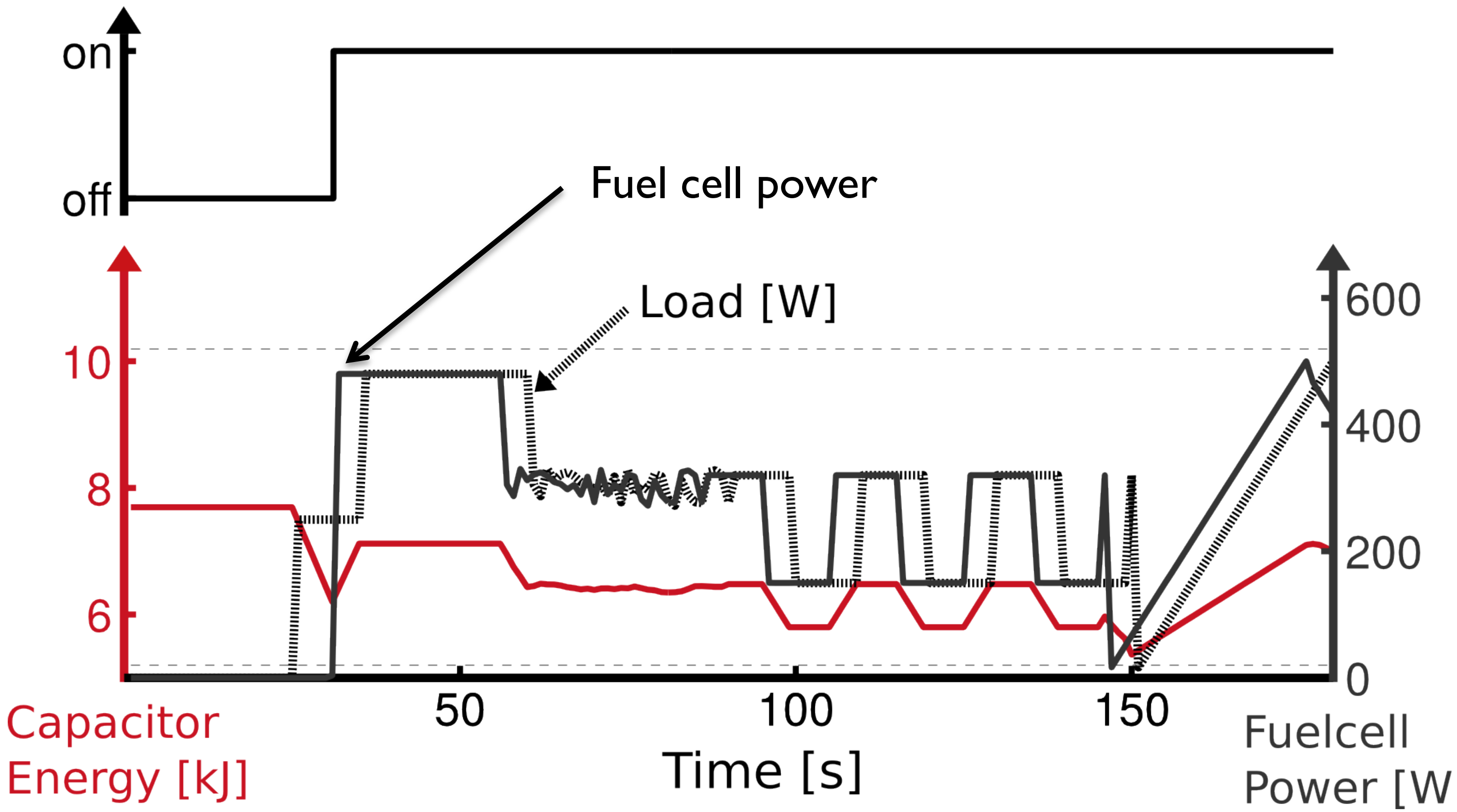
- Fuel cell power varied continuously, but discontinuity in losses
- Super-capacitor balances power

Control objectives:

- Meet (predicted) load profile
- Minimize losses (piecewise quadratic cost)
- Limit number of switchings to two per minute

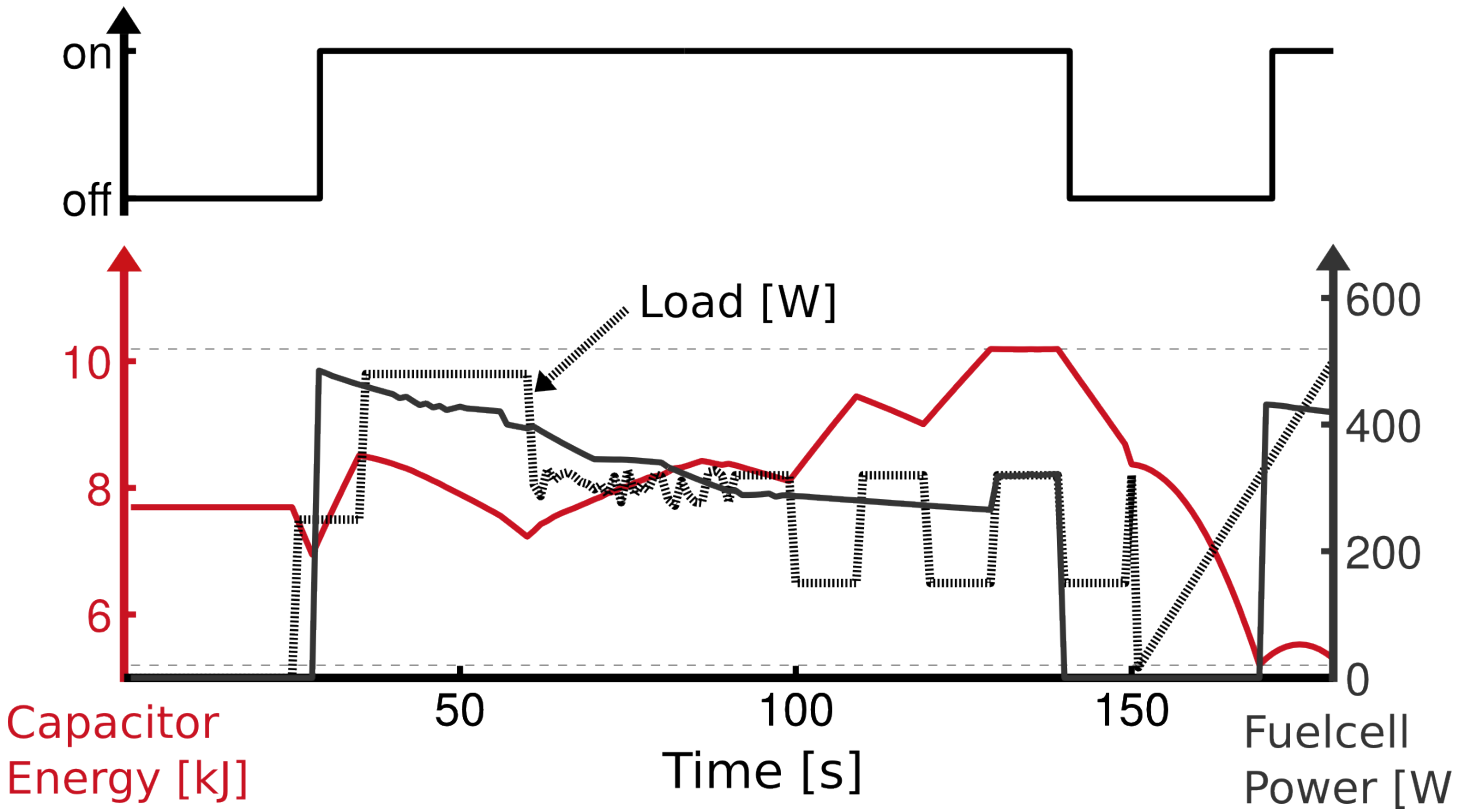
Application: Fuel Cell Power Management

Simulation profile for horizon 5s – losses: 31.5 kJ



Application: Fuel Cell Power Management

Simulation profile for horizon 60s – losses: 28.9 kJ



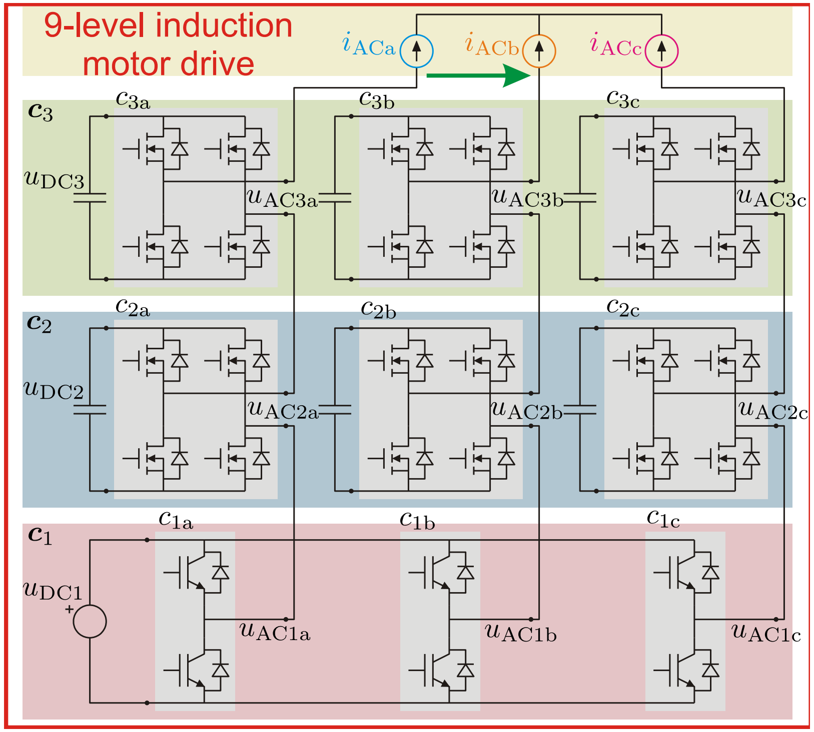
Losses reduced by 10% by long control horizon

Application: Multi-Level Inverters for Drives

- High efficiency and power quality

9-level induction motor drive:

- 6 capacitor voltages
- 3 motor currents
- 15 independent switches operated at frequency > 1kHz



- **Control objective:** track current reference
- MPC dramatically improves losses, distortion and transient behavior [Geyer et al., 2011]

Needs ultra-fast solver to compute switch positions in real-time

Embedded Solvers for Hybrid MPC

- **Main idea:** use fast convex sub-problem solvers + branch-and-bound
- On FPGA, 1 MHz per sub-problem possible [Jerez et al., 2013]

→ 1000 convex problems per millisecond – powerful decision making!

- **In this talk:**
  - Fast code-generated interior point solvers to cover general problems
  - Problem relaxations: exploit receding horizon + feedback in MPC
  - Pre-processing: detect infeasibility or sub-optimality without solving convex sub-problem

Goal: solver for hybrid MPC on embedded platforms

Outline

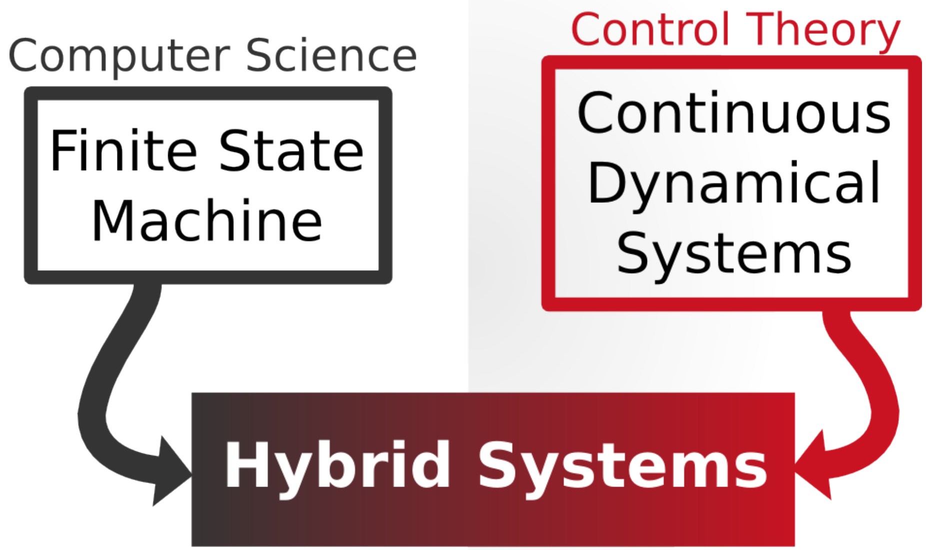
1. MLD models & hybrid MPC
2. Solution of mixed-integer programs via branch-and-bound
   - Standard branch & bound
   - Multistage problems & FORCES
3. Complexity reduction for embedded solvers
   - Rounding & branching strategy
   - Relaxations of optimality & feasibility
   - Pre-computations on binary constraints
4. Summary

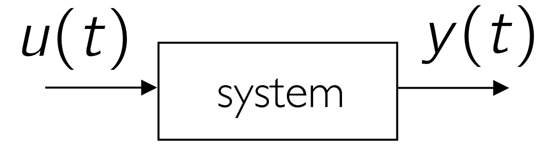
Outline

1. MLD models & hybrid MPC
2. Solution of mixed-integer programs via branch-and-bound
   - Standard branch & bound
   - Multistage problems & FORCES
3. Complexity reduction for embedded solvers
   - Rounding & branching strategy
   - Relaxations of optimality & feasibility
   - Pre-computations on binary constraints
4. Summary

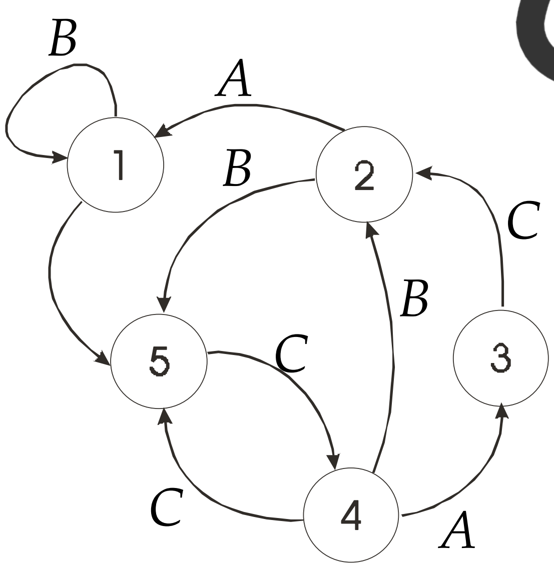
Hybrid Systems

$$X = \{1, 2, 3, 4, 5\}$$
$$U = \{A, B, C\}$$





$$x \in \mathbf{R}^n$$
$$u \in \mathbf{R}^m$$
$$y \in \mathbf{R}^p$$



$$x(k+1) = f(x(k), u(k))$$
$$y(k) = g(x(k), u(k))$$

Hybrid Systems with Affine Dynamics

- **Descriptive** enough to capture system behavior
  - continuous dynamics (physical laws)
  - logic components (switches, automata)
  - interconnection between logic and dynamics

- **Simple** enough for analysis and synthesis
  - controllability, observability, reachability
  - controller / filter design
  - stability & constraint satisfaction

Mixed Logic Dynamical (MLD) Models

- Discrete time linear dynamics and logic can be combined into Mixed Logic Dynamical (MLD) form [Bemporad & Morari, 1999]

$$
\begin{aligned}
x_{k+1} &= Ax_k + B_u u_k + B_w w_k + B_{aff} \\
y_k &= Cx_k + D_u u_k + D_w w_k + D_{aff} \\
E_x x_k &+ E_u u_k + E_w w_k \leq E_{aff}
\end{aligned}
$$

$$
\begin{aligned}
x &\in \mathbf{R}^{n_c} \times \{0,1\}^{n_b} \\
u &\in \mathbf{R}^{m_c} \times \{0,1\}^{m_b} \\
y &\in \mathbf{R}^{p_c} \times \{0,1\}^{p_b} \\
w &\in \mathbf{R}^{q_c} \times \{0,1\}^{q_b}
\end{aligned}
$$

- Mature modeling tools exists, e.g. HYSDEL [Torrisi & Bemporad, 2004]
- Equivalent to PWA, linear complementarity, max-min-plus-scaling

Hybrid MPC Problems with MLD Dynamics

$$\min_{x,u,w} \sum_{k=0}^{N-1} x_k^T Q x_k^T + u_k^T R u_k + x_N^T P x_N$$

$$\text{s.t. } x_0 = x$$

$$x_{k+1} = A x_k + B_u u_k + B_w w_k + B_{aff} \qquad x \in \mathbf{R}^{n_c} \times \{0,1\}^{n_b}$$

$$E_x x_k + E_u u_k + E_w w_k \leq E_{aff} \qquad u \in \mathbf{R}^{m_c} \times \{0,1\}^{m_b}$$

$$x_k \in \mathcal{X}_k, \quad x_N \in \mathcal{X}_f, \quad u_k \in \mathcal{U}_k \qquad w \in \mathbf{R}^{q_c} \times \{0,1\}^{q_b}$$

- Optimization problem is mixed-integer QP (NP-hard)
- Desktop software often solves medium-scale problems efficiently

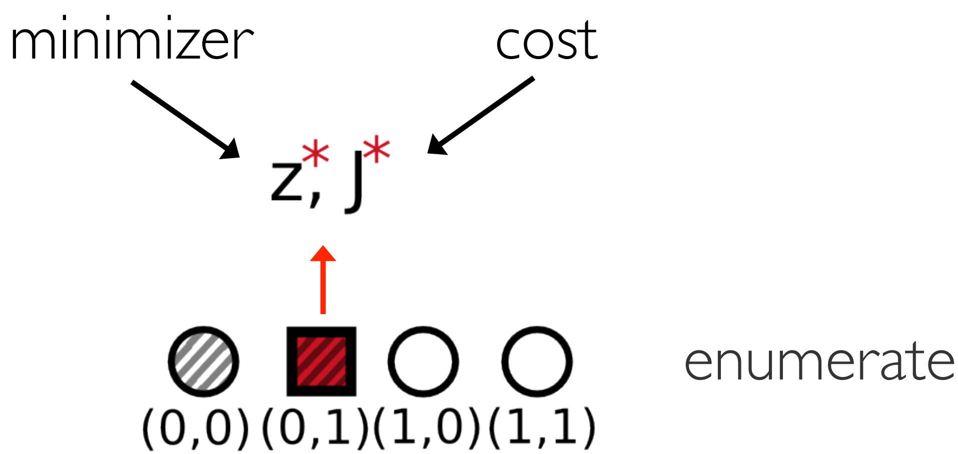
Embedded solvers exist only for small-scale problems (explicit)

Outline

1. MLD models & hybrid MPC
2. Solution of mixed-integer programs via branch-and-bound
   - Standard branch & bound
   - Multistage problems & FORCES
3. Complexity reduction for embedded solvers
   - Rounding & branching strategy
   - Relaxations of optimality & feasibility
   - Pre-computations on binary constraints
4. Summary

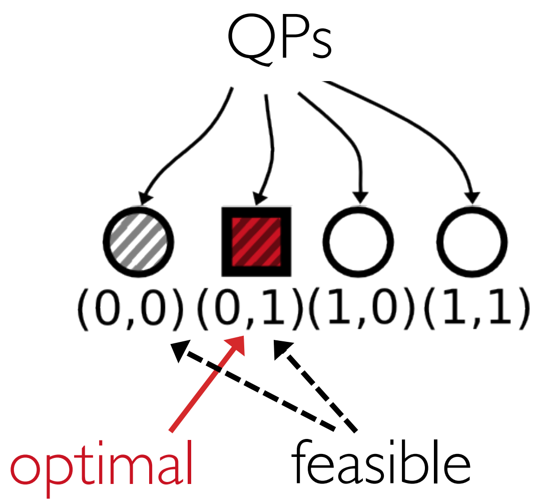
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities



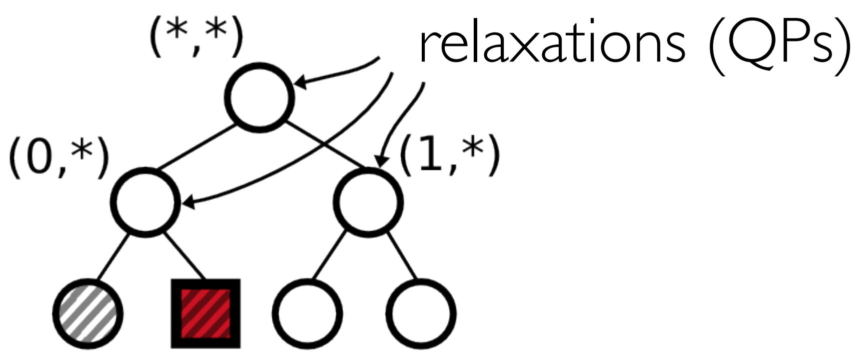
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities



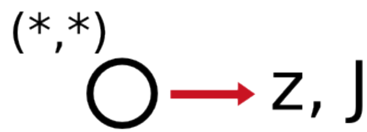
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



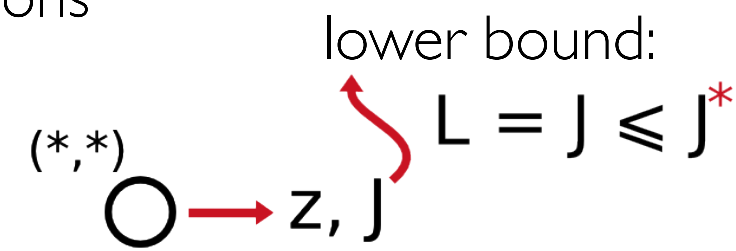
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



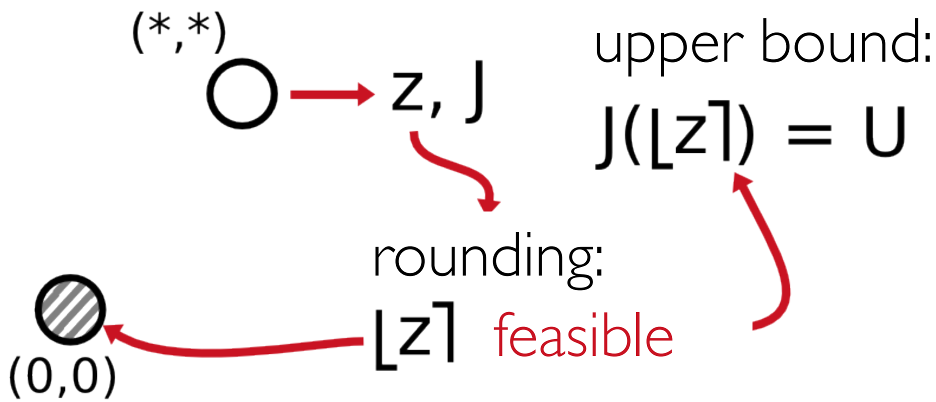
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



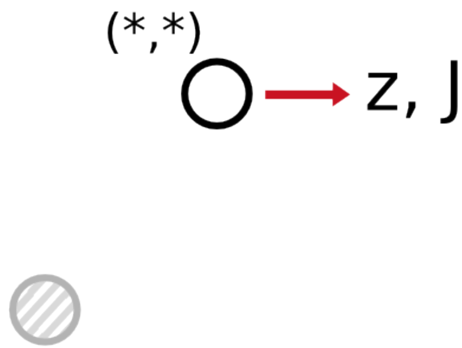
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



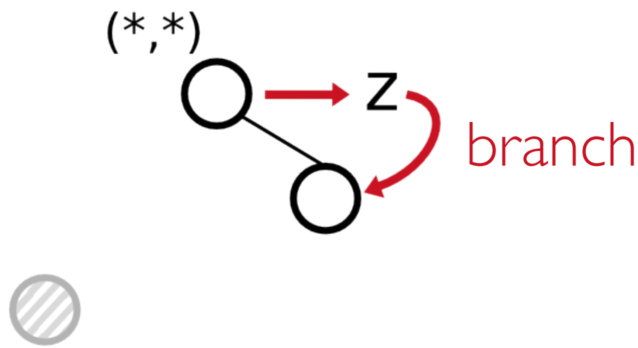
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



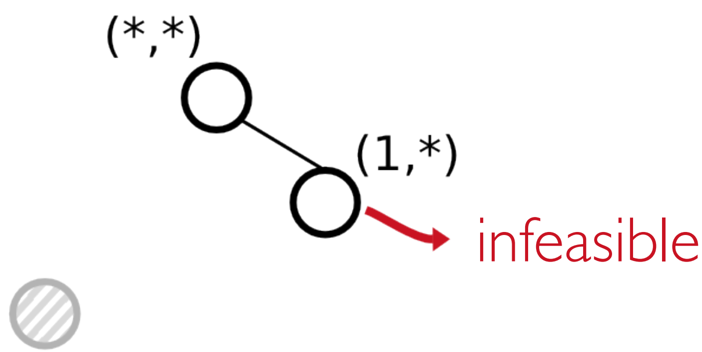
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



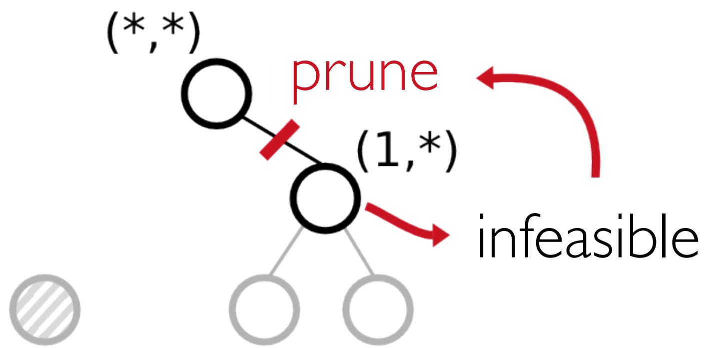
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



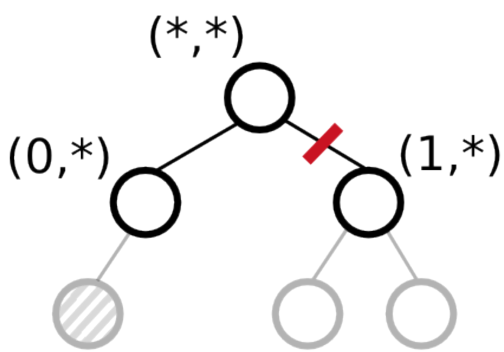
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



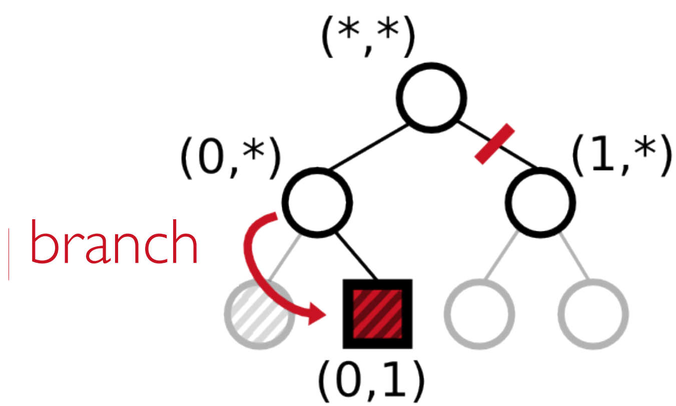
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



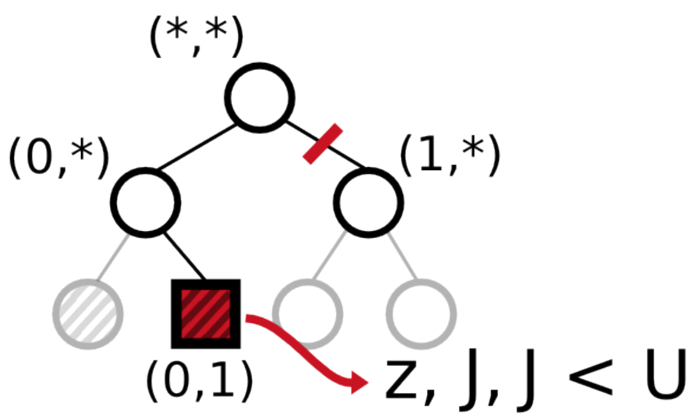
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



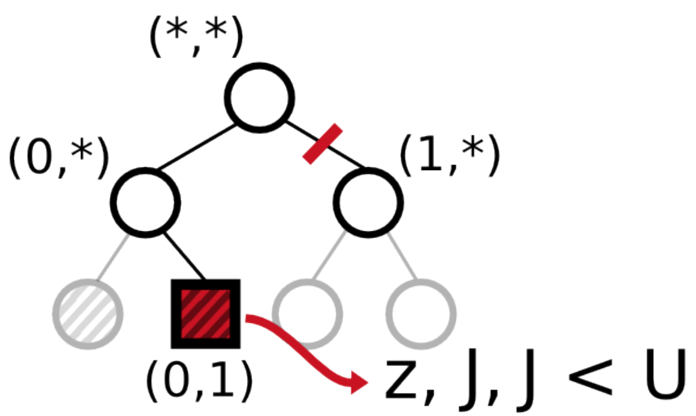
Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



Standard branch-and-bound

- Find optimal solution to MIQP without enumerating all possibilities
- Exploit bounds on optimal cost obtained from relaxations and integer-feasible solutions



- B&B code size: ~100 lines of C code

Key ingredient: fast, simple low-level (QC)QP solver

Multi-stage Convex QCQPs

minimize	$\sum_{i=1}^{N} \frac{1}{2} v_i^T H_i v_i + f_i^T v_i$	separable objective
subject to	$\underline{v}_i \leq v_i \leq \bar{v}_i$	upper/lower bounds
	$A_i v_i \leq b_i$	affine inequalities
	$v_i^T Q_{i,j} v_i + l_{i,j}^T v_i \leq r_{i,j}$	quadratic constraints
	$C_i v_i + D_{i+1} v_{i+1} = c_i$	inter-stage equalities

- Structure allows for significant speedups
- Hybrid MPC sub-problems fit this problem structure

Idea: use code-generated convex solver for B&B sub-problems

FORCES Exploits the Multistage Structure

- Main computation in interior point methods: solve linear system

Multistage QCQP

$$\min_{\{v_i\}} \sum_{i=0}^{N} l_i(v_i)$$
$$\text{s.t. } g_i(v_i) \leq 0$$
$$L_i(v_{i-1}, v_i) = 0$$

► **Search Direction Computation**

$$\begin{bmatrix} H & A^T & G^T & 0 \\ A & 0 & 0 & 0 \\ G & 0 & 0 & I \\ 0 & 0 & S & Z \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta y \\ \Delta z \\ \Delta s \end{bmatrix} = \begin{bmatrix} r_S \\ r_E \\ r_I \\ r_C \end{bmatrix}$$

► **Pos. Def. System**

$$Y \Delta y = b$$
$$Y = A\Phi^{-1}A^T$$
$$\Phi = H + G^T S Z^{-1} G$$

1. Compute Y (~80% of cost), 2. factor $Y = LL^T$ (~20%), 3. fwd./bkwd. solve

- Exploit block-wise structure in KKT system

Coefficient Matrix Y

$$\begin{bmatrix} Y_{0,0} & Y_{0,1} & 0 & \cdots & 0 \\ Y_{0,1}^T & Y_{1,1} & Y_{1,2} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & Y_{N-1,N} \\ 0 & 0 & \cdots & Y_{N-1,N}^T & Y_{N,N} \end{bmatrix}$$

◄ **New Method for Computing Y**

1	$L_i = \text{chol}(\Phi_i)$	$2/3r^3$
2	$V_i = C_i / L_i^T$	r^3
3	$W_i = D_i / L_i^T$	r^3
4	$Y_{i,i} = V_{i-1}^T V_{i-1}$	r^3
5	$+W_i^T W_i$	r^3
6	$Y_{i,i+1} = W_i V_i^T$	$2r^3$
	Total flops	$20/3r^3$

r: block size

- ✓ Saves $2r^3$ flops w.r.t. literature
- ✓ Cache efficient
- ✓ Numerically robust
- ✓ Parallelizable
- ✓ Enables further structure exploitation

Fine-Grained Structure Exploitation

- Additional structure exploitation possible for special cases:

Objective / Constraints

Theoretical speedups compared to base case	$c_i^T v_i$	$v_i^T Q v_i$, Q diag.	$v_i^T Q v_i$, Q dense
$\underline{v} \leq v_i \leq \bar{v}$	9.3x	9.3x	1.4x
$F v_i \leq f_i$	1.0x	1.0x	1.0x
$v_i^T M v_i \leq r$, M diag.	6.7x	6.7x	1.4x
$v_i^T M v_i \leq r$, M dense	1.4x	1.4x	1.4x (1.8x if M=Q)

- Example for typical MPC problem:
  - Stages 0...N-1: Q, R diagonal
  - Stage N: P dense

  → **~75% complexity reduction**

$$\begin{aligned} \min \quad & x_N^T P x_N + \textstyle\sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \\ \text{s.t.} \quad & x_0 = x, \;\; x_{i+1} = A x_i + B u_i \\ & \underline{x} \leq x_i \leq \bar{x}, \;\; \underline{u} \leq u_i \leq \bar{u}, \\ & x_N^T P x_N \leq \alpha \end{aligned}$$

[Domahidi et al., CDC 2012]

Structure exploitation can be automatized by code generation

The FORCES Code Generator

Multistage QCQP

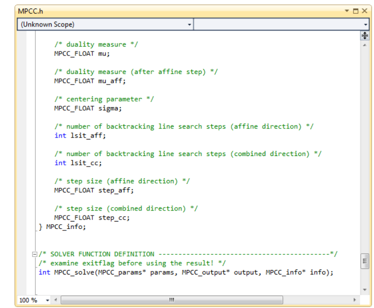
$$
\begin{aligned}
\min \quad & \sum_{i=1}^{N} \tfrac{1}{2} v_i^T H_i v_i + f_i^T v_i \\
\text{s.t.} \quad & \underline{z}_i \le v_i \le \bar{v}_i \\
& A_i v_i \le b_i \\
& v_i^T Q_{i,j} v_i + l_{i,j}^T v_i \le r_{i,j} \\
& C_i v_i + D_{i+1} v_{i+1} = c_i
\end{aligned}
$$

?

Embedded Hardware

Generated Code

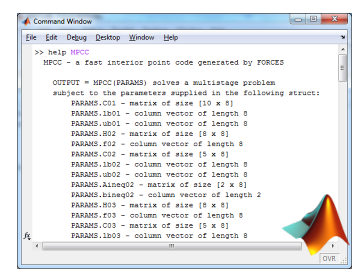
Solver (ANSI-C)



```
MPCC.h
(Unknown Scope)

    /* duality measure */
    MPCC_FLOAT mu;

    /* duality measure (after affine step) */
    MPCC_FLOAT mu_aff;

    /* centering parameter */
    MPCC_FLOAT sigma;

    /* number of backtracking line search steps (affine direction) */
    int lsit_aff;

    /* number of backtracking line search steps (combined direction) */
    int lsit_cc;

    /* step size (affine direction) */
    MPCC_FLOAT step_aff;

    /* step size (combined direction) */
    MPCC_FLOAT step_cc;
} MPCC_info;


/* SOLVER FUNCTION DEFINITION -------------------------------------------*/
/* examine exitflag before using the result! */
int MPCC_solve(MPCC_params* params, MPCC_output* output, MPCC_info* info);
```

```
solver.h
solver.c
solver.m
solvermex.c
makemex
```

MATLAB MEX interface for rapid prototyping



```
Command Window
File Edit Debug Desktop Window Help
>> help MPCC
  MPCC - a fast interior point code generated by FORCES

    OUTPUT = MPCC(PARAMS) solves a multistage problem
    subject to the parameters supplied in the following struct:
        PARAMS.C01 - matrix of size [10 x 8]
        PARAMS.lb01 - column vector of length 8
        PARAMS.ub01 - column vector of length 8
        PARAMS.H02 - matrix of size [8 x 8]
        PARAMS.f02 - column vector of length 8
        PARAMS.C02 - matrix of size [5 x 8]
        PARAMS.lb02 - column vector of length 8
        PARAMS.ub02 - column vector of length 8
        PARAMS.Aineq02 - matrix of size [2 x 8]
        PARAMS.bineq02 - column vector of length 2
        PARAMS.H03 - matrix of size [8 x 8]
        PARAMS.f03 - column vector of length 8
        PARAMS.C03 - matrix of size [5 x 8]
        PARAMS.lb03 - column vector of length 8
```

Problem description

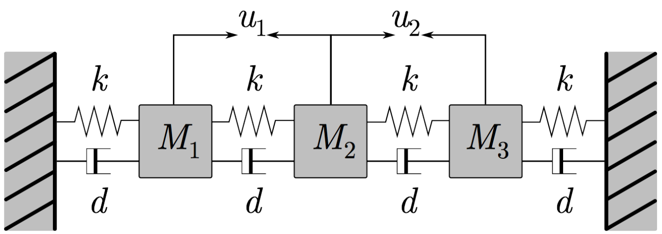
```
stage = MultiStageProblem(N+1);
for i = 1:N+1
% dimensions
stages(i).dims.n = 10;
stages(i).dims.r = 5;
stages(i).dims.lb = 3;

% cost
stages(i).cost.H = Hi;
stages(i).cost.f = fi;

% inequalities
stages(i).ineq.b.lbidx = 3:5;
stages(i).ineq.b.lb = zeros(3,1);

% equalities
stages(i).eq.C = Ci;
stages(i).eq.c = ci;
stages(i).eq.D = Di;
end
generateCode(stages);
```

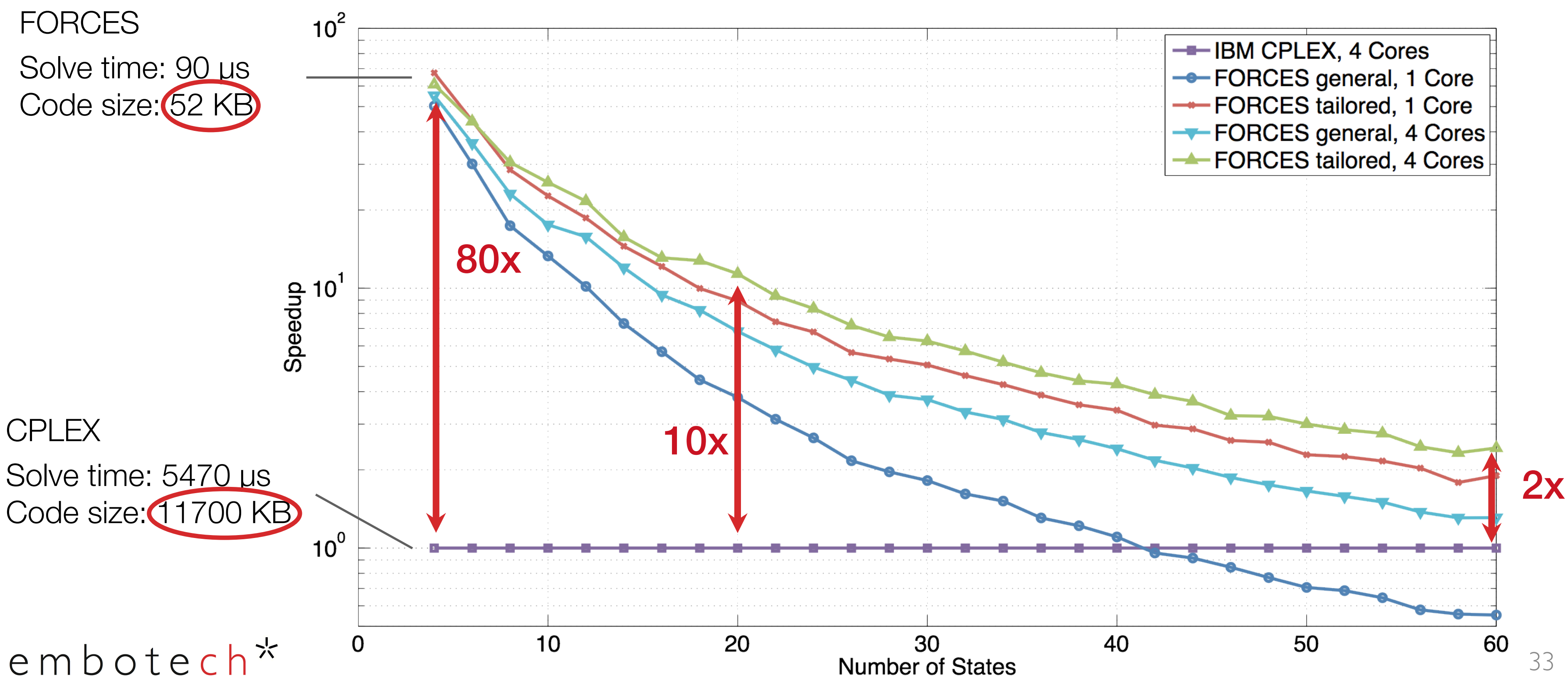
FORCES

- C code generation of primal-dual Mehrotra interior point solvers
- LPs, QPs, QCQPs
- Parametric problems
- Multi-core platforms
- Library-free
- Available: forces.ethz.ch

Speedups Compared to IBM CPLEX

- Standard MPC problem for oscillating chain of masses (on Intel i5 @3.1 GHz)
- CPLEX N/A on embedded systems





Simple B&B Strategy with FORCES

- Generate a solver for QCQP with parametric lower- and upper bounds:

$$
\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{i=1}^{N} \frac{1}{2} v_i^T H_i v_i + f_i^T v_i \\
\text{subject to} \quad & \underline{v}_i \le v_i \le \bar{v}_i \\
& A_i v_i \le b_i \\
& v_i^T Q_{i,j} v_i + l_{i,j}^T v_i \le r_{i,j} \\
& C_i v_i + D_{i+1} v_{i+1} = c_i
\end{aligned}
$$

Change in each B&B sub-problem

- Relaxation: $0 \le v_i \le 1$ for binary variables
- Set $\underline{v}_{i,j} = \bar{v}_{i,j}$ to 0 or 1 to fix variable j in stage i

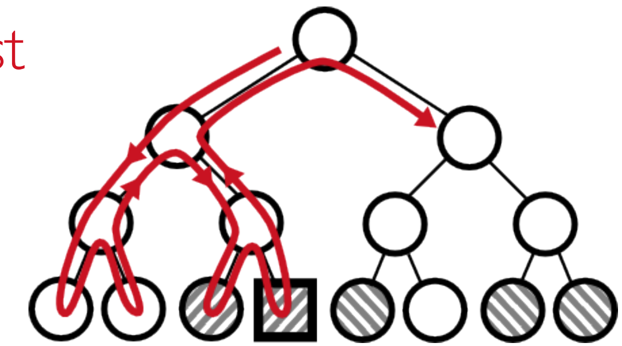
→ Low-footprint MI-QCQP solver for hybrid MPC

Outline

1. MLD models & hybrid MPC
2. Solution of mixed-integer programs via branch-and-bound
    - Standard branch & bound
    - Multistage problems & FORCES
3. Complexity reduction for embedded solvers
    - Rounding & branching strategy
    - Relaxations of optimality & feasibility
    - Pre-computations on binary constraints
4. Summary

Branch-and-bound: Branching Strategy

- Branch-and-bound tree is explored depth-first
  - Likely to find feasible solutions early
  - Low memory complexity (linear in #binaries)



- Use stage-in-order heuristic (fix in order $i = 1, 2, \ldots, N$)
  - Motivated by receding horizon policy – fix early stages first

- In current stage, branch on most ambiguous relaxed variable, i.e. the one closest to 0.5 [Boyd & Mattingley, EE364b notes]

Branch-and-bound: Rounding Schemes

- Nearest-neighbor: computationally efficient, solution can be infeasible

- Combinatorial Integral Approximation [Sager et al. 2014]

$$\tilde{\delta} \triangleq \arg\min_{\delta \in \Omega} \max_{k=0,\ldots,N} \max_{j} \left| \sum_{l=0}^{k} \left(\delta_{l,j}^{\star} - \delta_{l,j}\right) \right|$$

  - Need to solve MILP, but theoretical guarantees on approx. quality

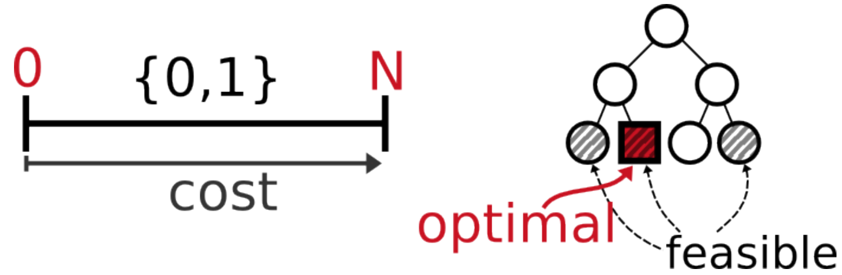
- Sum-up-rounding: explicit solution of CIA if $\Omega \equiv \{0,1\}^n$ [Sager et al. 2012]

$$\tilde{\delta}_{k,j} \triangleq \begin{cases} 1 & \text{if } \sum_{l=0}^{k} \delta_{l,j}^{\star} - \sum_{l=0}^{k-1} \tilde{\delta}_{l,j} \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Outline

1. MLD models & hybrid MPC
2. Solution of mixed-integer programs via branch-and-bound
   - Standard branch & bound
   - Multistage problems & FORCES
3. Complexity reduction for embedded solvers
   - Rounding & branching strategy
   - Relaxations of optimality & feasibility
   - Pre-computations on binary constraints
4. Summary

Relaxations

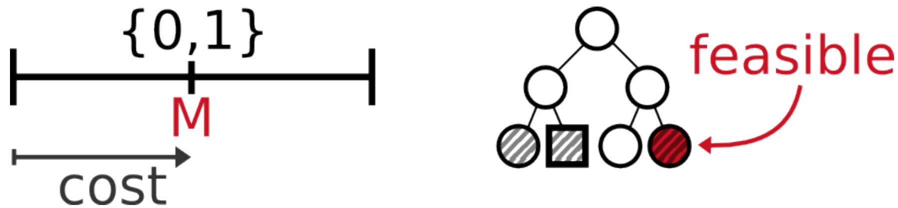


- Only first control action is applied in a receding horizon scheme
  → Can relax later stages to improve solve time

- Relax integrality or optimality after M < N stages

Trade-off computation time for performance

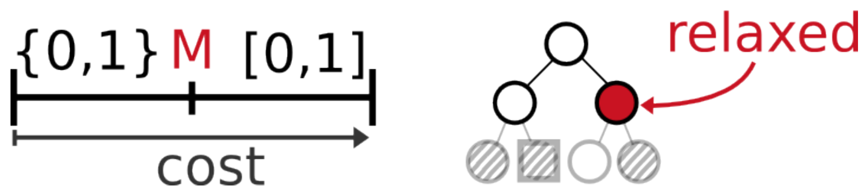
Relaxation of Optimality



- Relax optimality after M stages, preserve feasibility for all N stages
  - Often N-step feasible, M-step optimal solutions of sufficient quality for closed-loop control
  - Reduces complexity if feasible solutions can be found quickly

- Solution is feasible for original problem, but likely to be suboptimal

Relaxation of Integer Feasibility

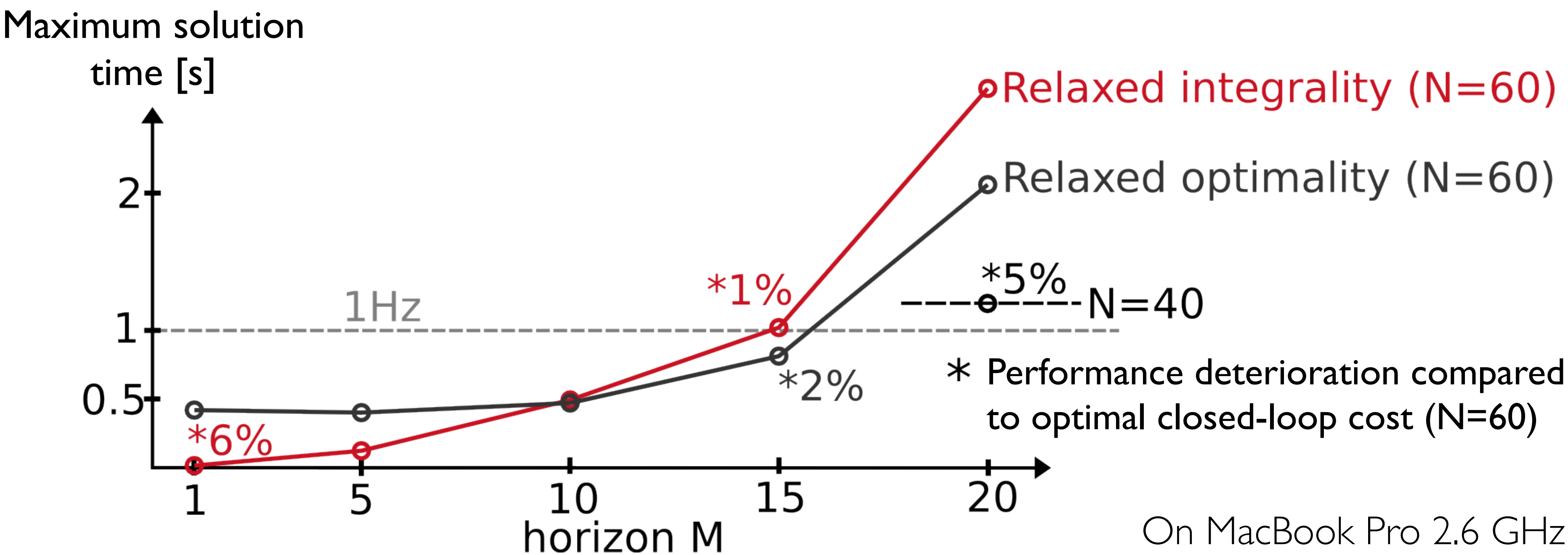


- Relax integer feasibility after M stages
  - Maintain benefits of long horizons for continuous dynamics
  - First control move likely to provide good performance

- Effective reduction of #binaries

Numerical Results: Fuel Cell Control

- Hybrid MPC problem with 60 binary and 242 continuous variables

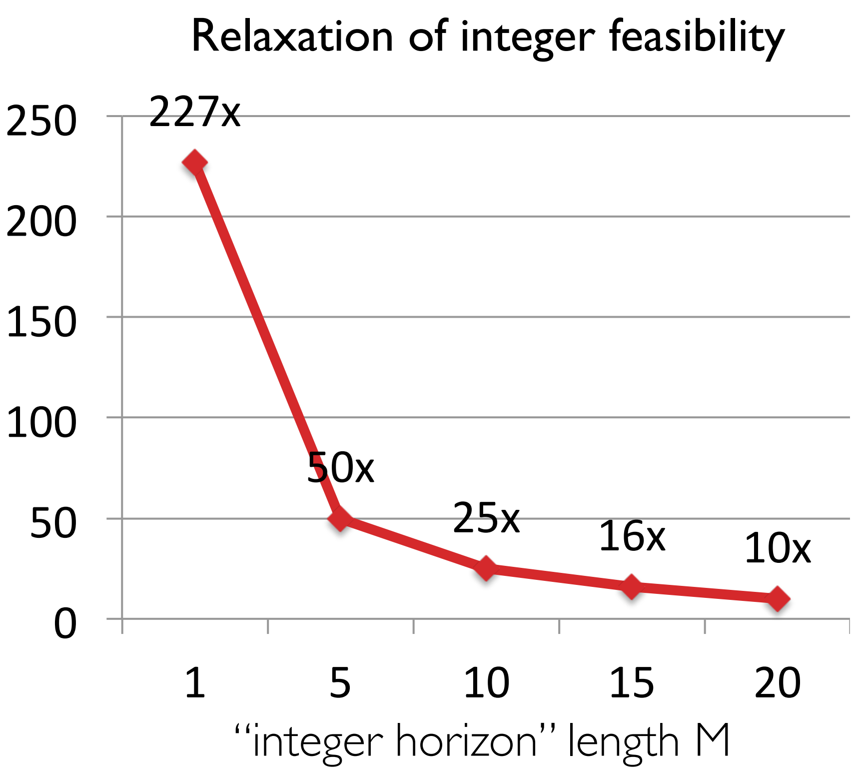


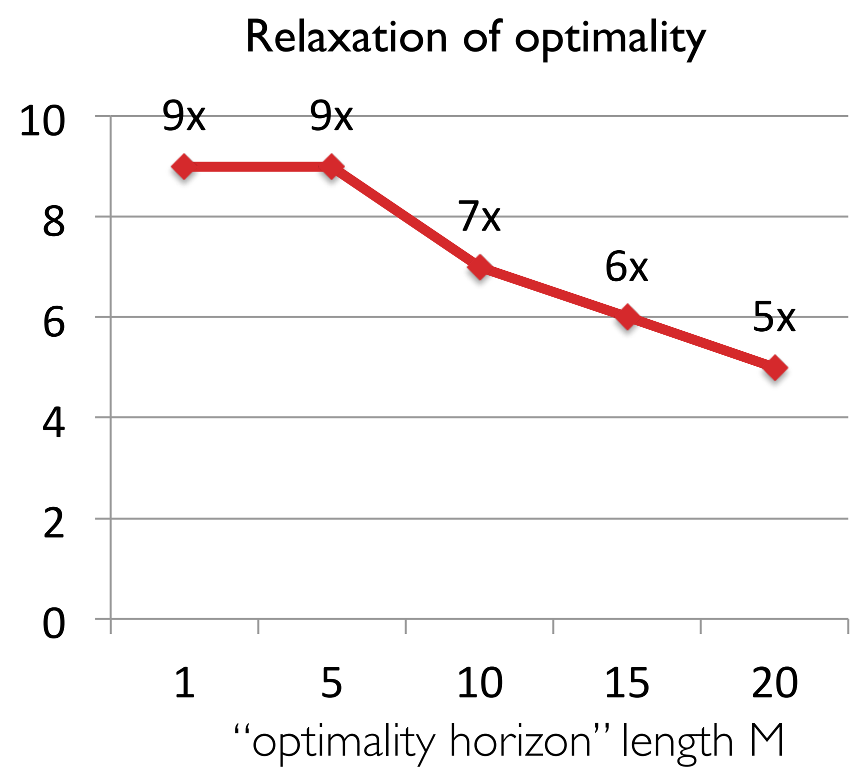
On MacBook Pro 2.6 GHz

- Sampling time of 1s met with 1% performance deterioration
- Max. 208 QPs solved

Speedups Compared to IBM CPLEX

- Speedup of median compute time w.r.t. CPLEX solving full problem:



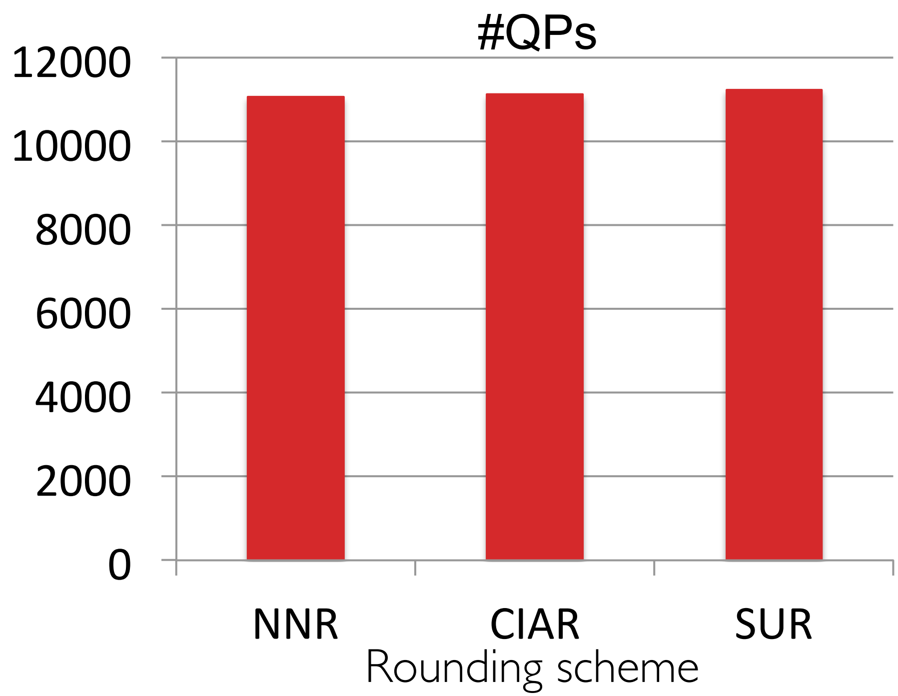


- In all cases with M>5: performance deterioration less than 2%

embotech*

Numerical Results: Optimal Traffic Control

- Horizon length N=5: 25 binary variables, 105 continuous variables
- Number of QPs applying standard approach without relaxations:



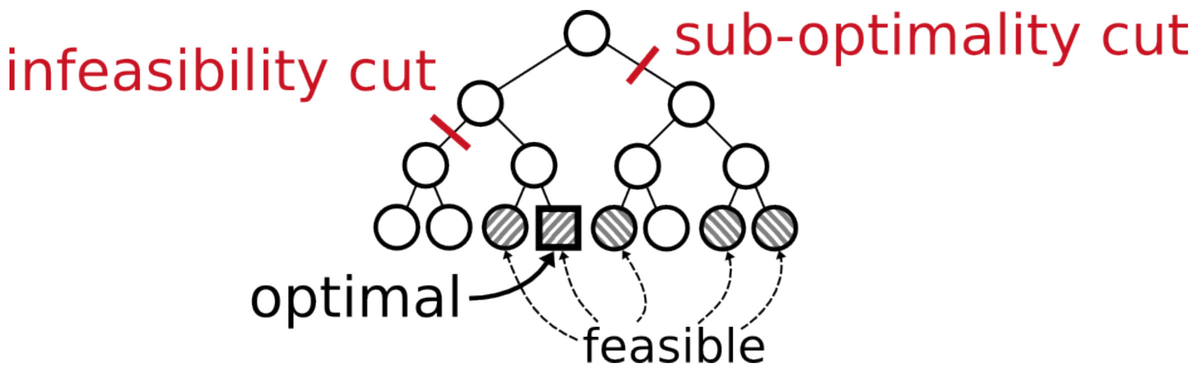
- Solution time ~20 seconds → too slow
- ~96% of time for "solving" infeasible sub-problems

Outline

1. MLD models & hybrid MPC
2. Solution of mixed-integer programs via branch-and-bound
   - Standard branch & bound
   - Multistage problems & FORCES
3. Complexity reduction for embedded solvers
   - Rounding & branching strategy
   - Relaxations of optimality & feasibility
   - Pre-computations on binary constraints
4. Summary

Pre-processing: Infeasibility/Sub-optimality Pruning

- **Main idea:** exploit structure of constraints & cost on binaries to prune sub-trees without solving convex sub-problem
- Infeasibility pruning
- Sub-optimality pruning



Goal: Pre-compute at codegen for efficient pruning at run time

Pre-processing: Extended Constraint Functions

- Consider a constraint depending only on binaries:

$$g(\delta) \leq 0, \quad \delta \in \{0,1\}^{n_b}$$

- Each B&B node can be represented by a set $\mathcal{I} \subseteq 2^{\{0,1\}^{n_b}}$ of possible binaries
- Goal: detect infeasibility of $g(\delta)$ for all $\delta \in \mathcal{I}$ efficiently

→ Extended constraint function (ECF):

$$g_e(\mathcal{I}) \triangleq \min_{\delta} \quad g(\delta) \quad \text{s.t.} \quad \delta \in \mathcal{I}$$

with property: $g_e(\mathcal{I}) > 0 \Rightarrow g(\delta) > 0 \; \forall \delta \in \mathcal{I}$
(sufficient condition for infeasibility)

Pre-processing: Extended Constraint Functions

$$
\begin{aligned}
g_e(\mathcal{I}) \triangleq \min_{\delta} \quad & g(\delta) \\
\text{s.t.} \quad & \delta \in \mathcal{I}
\end{aligned}
$$

- ECF is a parametric integer program
- Cheap to evaluate $g_e(\cdot)$ for
  - Constraint on #switchings (fuel cell, power electronics)
  - Exclusive logical constraints (traffic control, multilevel inverter)
  - Lookup table, decision tree (for small number of variables)

Infeasibility Pruning via ECFs

- Code for evaluating $g_e(\cdot)$ is generated at codegen time
- Evaluating $g_e(\cdot)$ is often more efficient than solving QP
- At each B&B node, evaluate $g_e(\mathcal{I})$ to test for infeasibility

- Example: constraint on #switchings (fuel cell)

$$g_e(\mathcal{I}) \triangleq \min_{\delta \in \mathcal{I}} \sum_{j=k-N_s+1}^{k} \delta_j - n_{switch}$$

window length (for past)

# of past switchings

  - Trivial evaluation: set relaxed binaries to zero & count fixed 1's

Use $g_e(\cdot)$ to prune infeasible sub-trees

embotech*

Sub-optimality Pruning via ECFs

- **Key assumption 1:** optimization problem has the form

$$\begin{aligned} \min_{z,\delta} \; & f(z,\delta) \\ \text{s.t.} \; & (z,\delta) \in \mathcal{C} \subseteq R^{n_c} \times \{0,1\}^{n_b} \\ & I_G^T \delta \leq 1 \end{aligned}$$

where I_G is the incidence matrix of the undirected graph $G \triangleq (V, E)$ modeling the dependency of binaries

- Models logical constraints that are exclusive
  Example traffic control: only certain combinations of signals on green
- Consequence: every feasible δ is an independent set
  (i.e. nodes are not connected by edges)

embotech*

Sub-optimality Pruning via ECFs

- **Key assumption 2:** cost does not increase when setting a 0 to 1, i.e.

  for all feasible $\delta, \hat{\delta} \in \{0,1\}^{n_b}: \ \|\delta\|_1 > \|\hat{\delta}\|_1 \Rightarrow h(\delta) \leq h(\hat{\delta})$

  where $h(\delta) \triangleq \begin{cases} \min_z \left\{ f(z,\delta) : (z,\delta) \in \mathcal{C}, I_G^T \delta \leq 1 \right\} & \text{if feasible} \\ +\infty & \text{otherwise} \end{cases}$

  Example traffic control: set as many signals to green as feasible

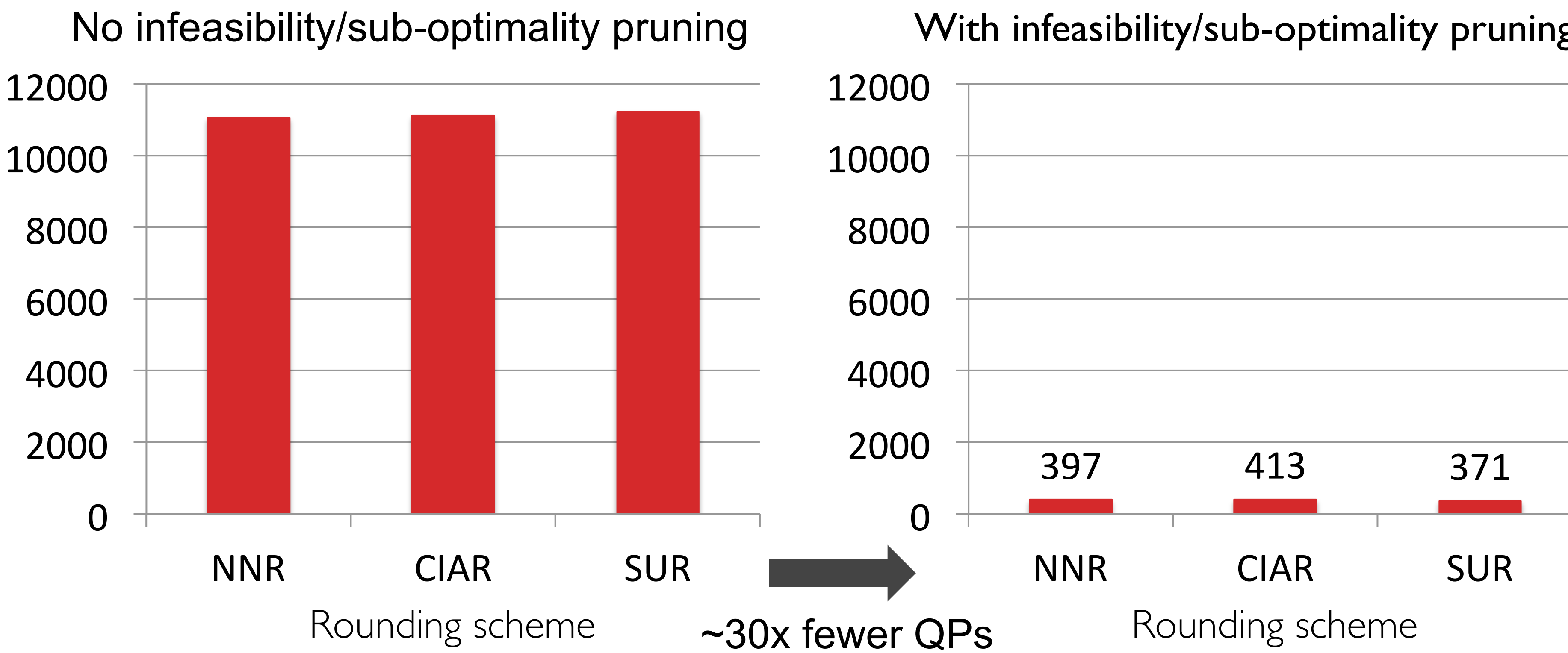
- Consequence: every optimal δ is a maximum independent set (i.e. no node can be added such that all nodes are not connected)

- Set of maximum independent sets can be pre-computed

Use $g_e(\cdot)$ to prune non-maximum (suboptimal) independent sets

Numerical Results: Optimal Traffic Control

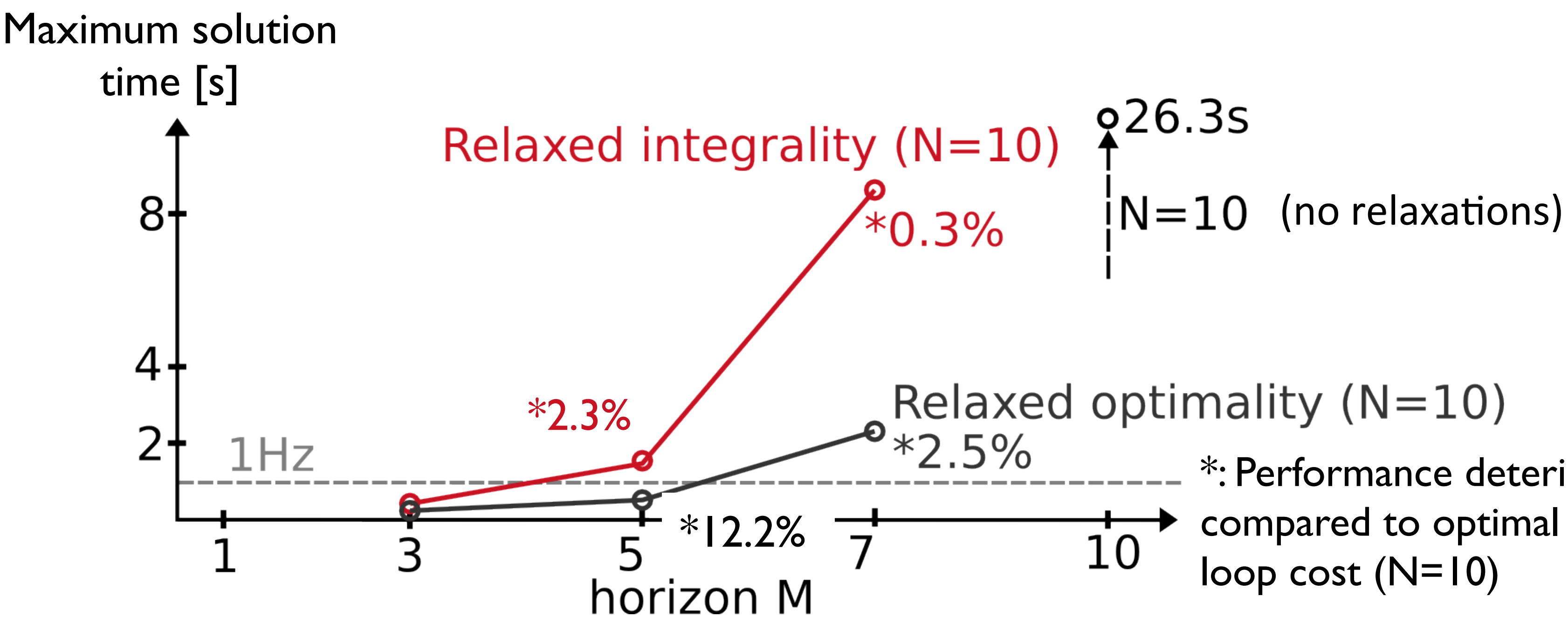
- For horizon length of 5: 25 binary variables, 105 continuous variables



~30x fewer QPs

Infeasibility & sub-optimality pruning drastically reduces #QPs

Numerical Results: Traffic Control



*: Performance deterioration compared to optimal closed-loop cost (N=10)

- Median computation times compared to CPLEX:
  - 30x slower for N=10 (full problem)
  - 3x slower for M=5 (2.3% performance loss)

Approaching speed of desktop solvers with embeddable code

Summary

- Many control problems need solution of MIPs on embedded systems
- First approach toward embedded solver for hybrid MPC:
  - Standard branch-and-bound + tailored, generated interior-point solver
  - Stage-in-order + most-ambiguous branching, nearest-neighbor rounding
  - Code size few tens of KB
- Pre-processing: detect infeasible/suboptimal nodes w/o solving QPs
  - Extended constraint functions can be generated at codegen time
  - Often much faster to evaluate than convex sub-problem

→ Simple embeddable solver approaching desktop performance

- Future work:
  - First-order sub-problem solvers (ADMM, gradient projection, ...)
  - Generate more tree-pruning cuts at code generation time