HOT STARTED NLP SOLVERS

Applications and Numerical Experiments

Travis C. Johnson and Andreas Wächter

Northwestern University

INFORMS Annual Meeting 2012

Outline

* Motivation
* Applications
* Algorithm
* Experiments

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & c(x) \geq 0 \\ & x \geq l \end{aligned}$$

$f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are sufficiently smooth

Big Picture

* Some applications require solution of **closely related NLPs** with slightly different data

* Examples:

  * Branch-and-bound for MINLP: Strong Branching and Diving

  * Online nonlinear model predictive control: external input

Big Picture

- MILP case:
  - after a bound change, dual simplex LP solver can rapidly solve new instance
    - Reason: an existing factorization of basis matrix can be used
    - May speed up MILP branch-and-bound significantly
- **Can we hot-start NLP solvers, re-using an existing factorization?**

Framework: SQP methods

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & \frac{1}{2} d^T W_k d + g_k^T d \\ \text{s.t.} \quad & A_k d + c_k = 0, \quad d \geq l_k \end{aligned}$$

* $W_k = \nabla^2 \mathcal{L}(x_k, \lambda_k)$
* $A_k = \nabla c(x_k)$
* depend on x_k, λ_k

* Usually: W and A change, use a QP solver's warmstart
  * Use previous active set as starting guess(but refactorize)

---

* Hot starts of an active set QP solver
  * make only g, c, l changes...
  * as long as W, A stay constant! (avoid recomputing internal matrix factorization--expensive!)

iQP Paradigm

* Big idea: Had to factorize at least one QP--can we reuse it later?

SOLVE

$$\begin{aligned} \min_{x,y} \quad & \frac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & Ad + c = 0 \\ & d \geq l \end{aligned}$$

(New W, A)

USING:

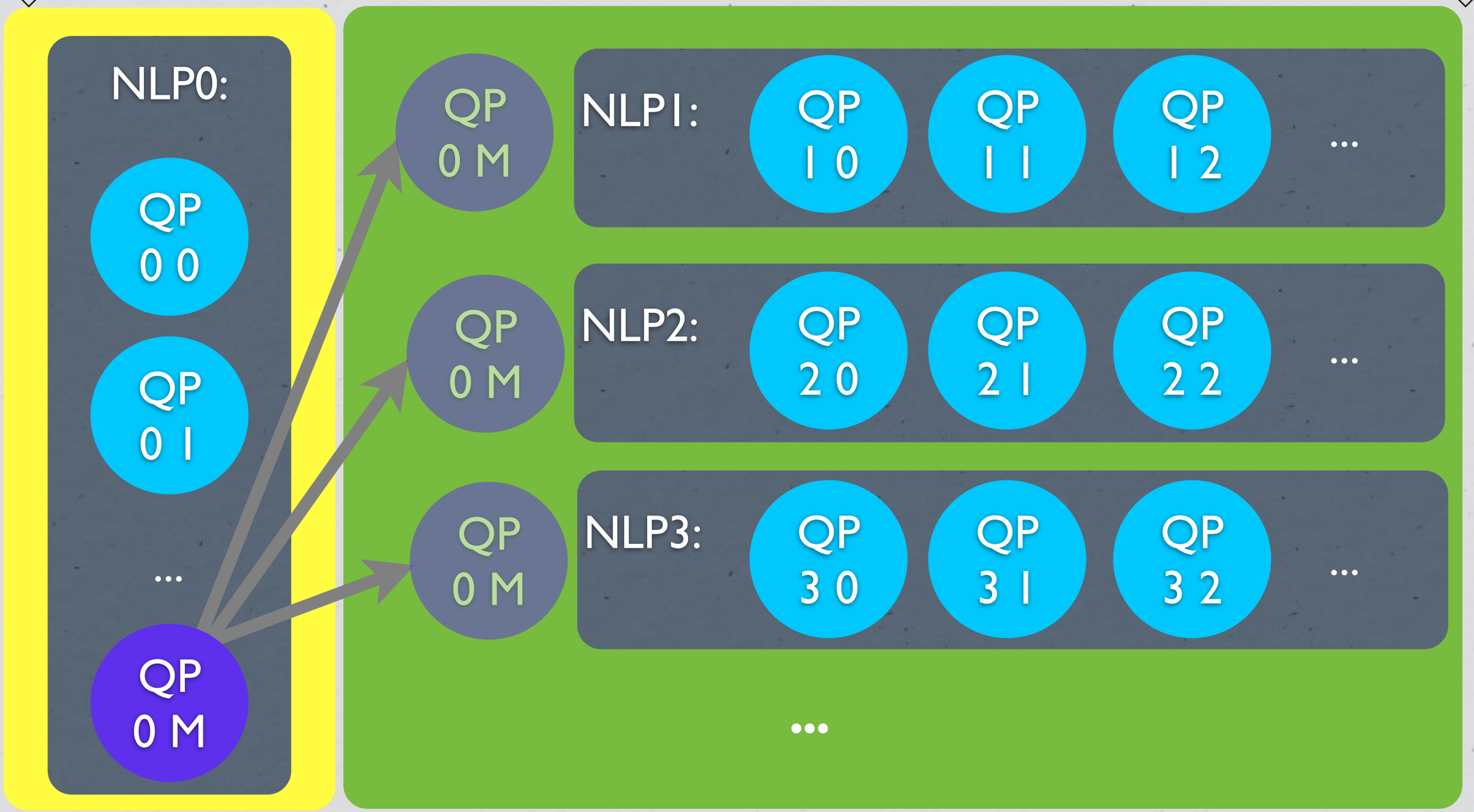
$$\begin{aligned} \min_{x,y} \quad & \frac{1}{2} d^T W_0 d + g^T d \\ \text{s.t.} \quad & A_0 d + c = 0 \\ & d \geq l \end{aligned}$$

With hotstarts and different g, c, l

(ie, factorization from this QP)

* Warm start: Use active set

iQP's Ideal Habitat



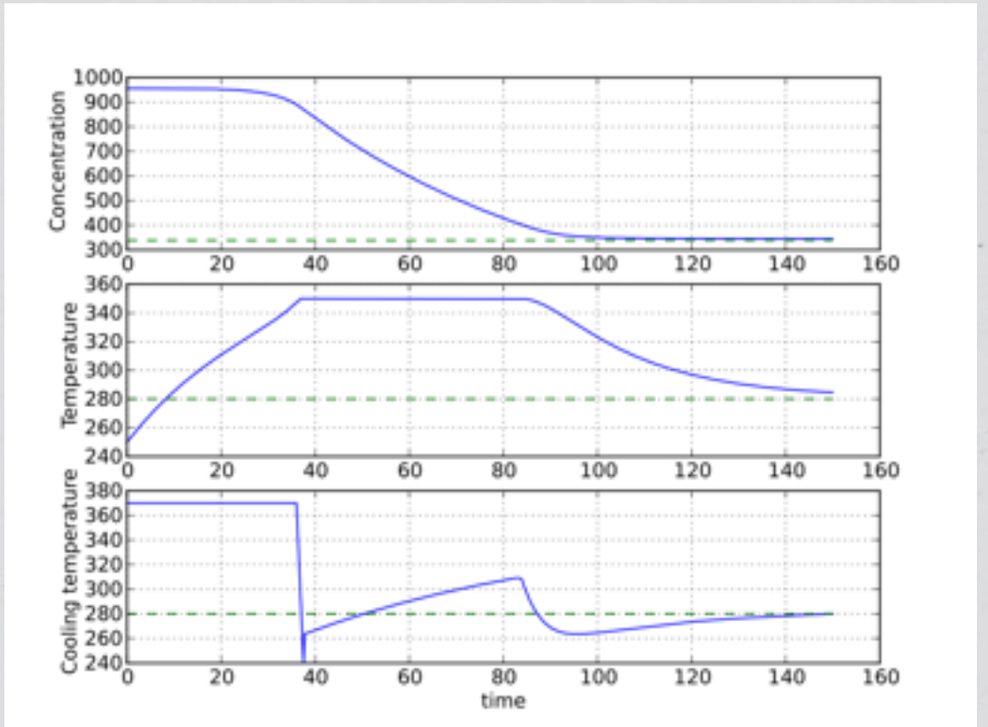
Online Nonlinear Model Predictive Control

$$\begin{aligned} \min \quad & f(x,u,p) \\ \text{s.t.} \quad & c(x,u,p) \geq 0 \\ & \dot{x} = h(x,u,p) \\ & x(t=t_0) = \tilde{x}_0 \end{aligned}$$

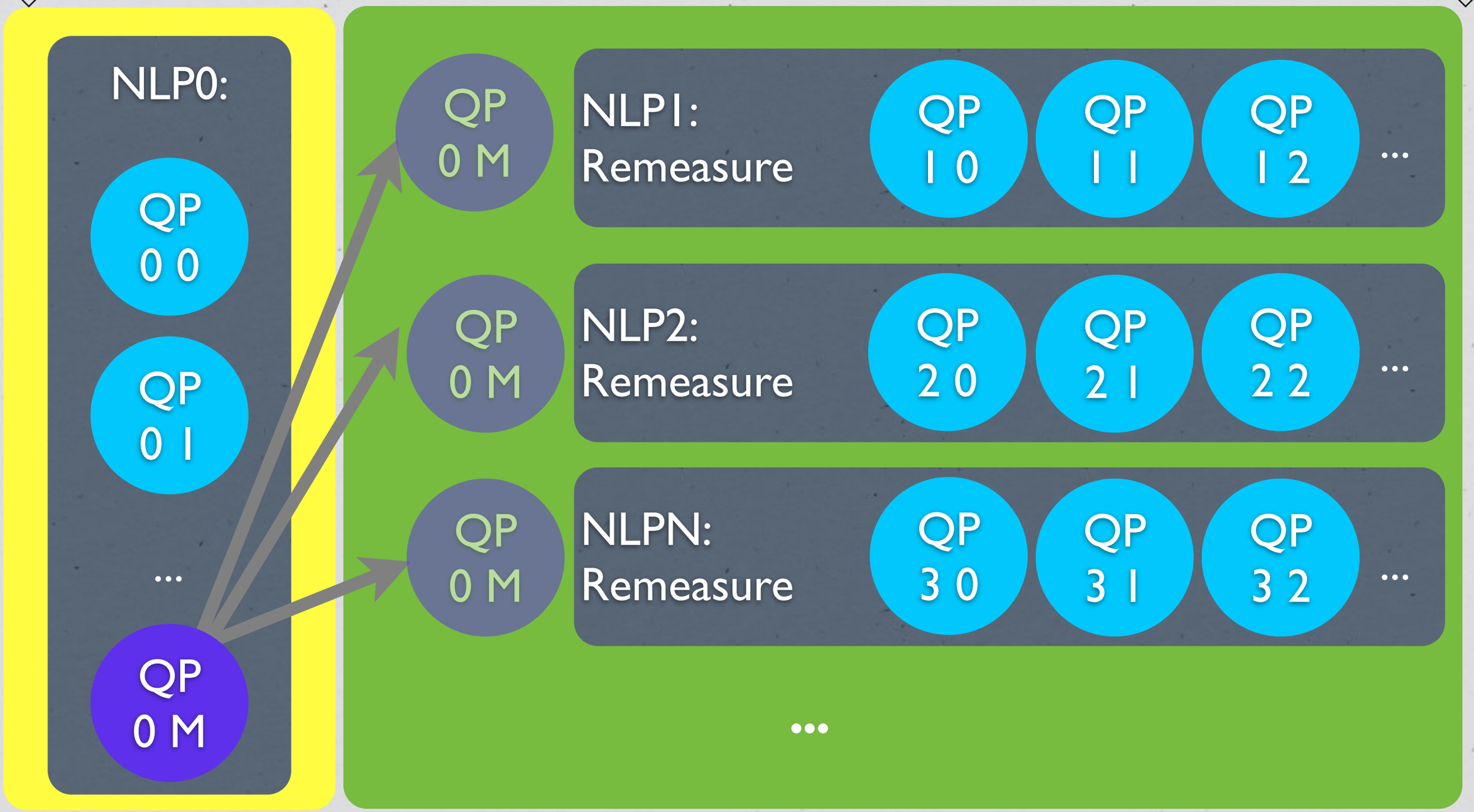
$$\Rightarrow$$

$$\begin{aligned} \min \quad & f(x,u,p) \\ \text{s.t.} \quad & c(x,u,p) \geq 0 \\ & x_{i+1} = x_i + (\Delta t) h(x,u,p) \\ & x(t=t_0) = \tilde{x}_0 \end{aligned}$$

* **Multiple** NLPs from:
  * changes in initial conditions
  * changes in external parameters



Nonlinear Model Predictive Control



Mixed Integer NLP & Branch-and-bound

$$\begin{aligned} \min \quad & f(x,y) \\ \text{s.t.} \quad & c(x,y) \geq 0 \\ & x \geq l \\ & y \in \{0,1\} \end{aligned}$$

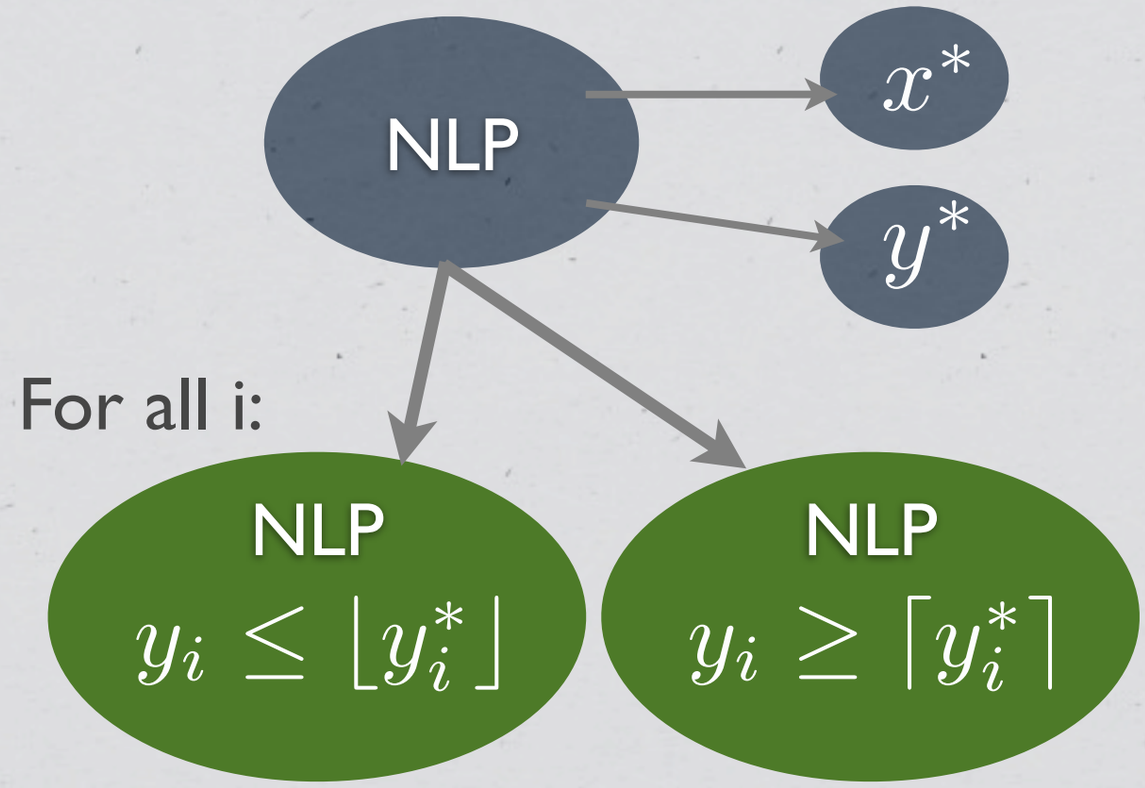
* Branch-and-bound
  * Strong Branching
  * Diving

MINLP Strong Branching

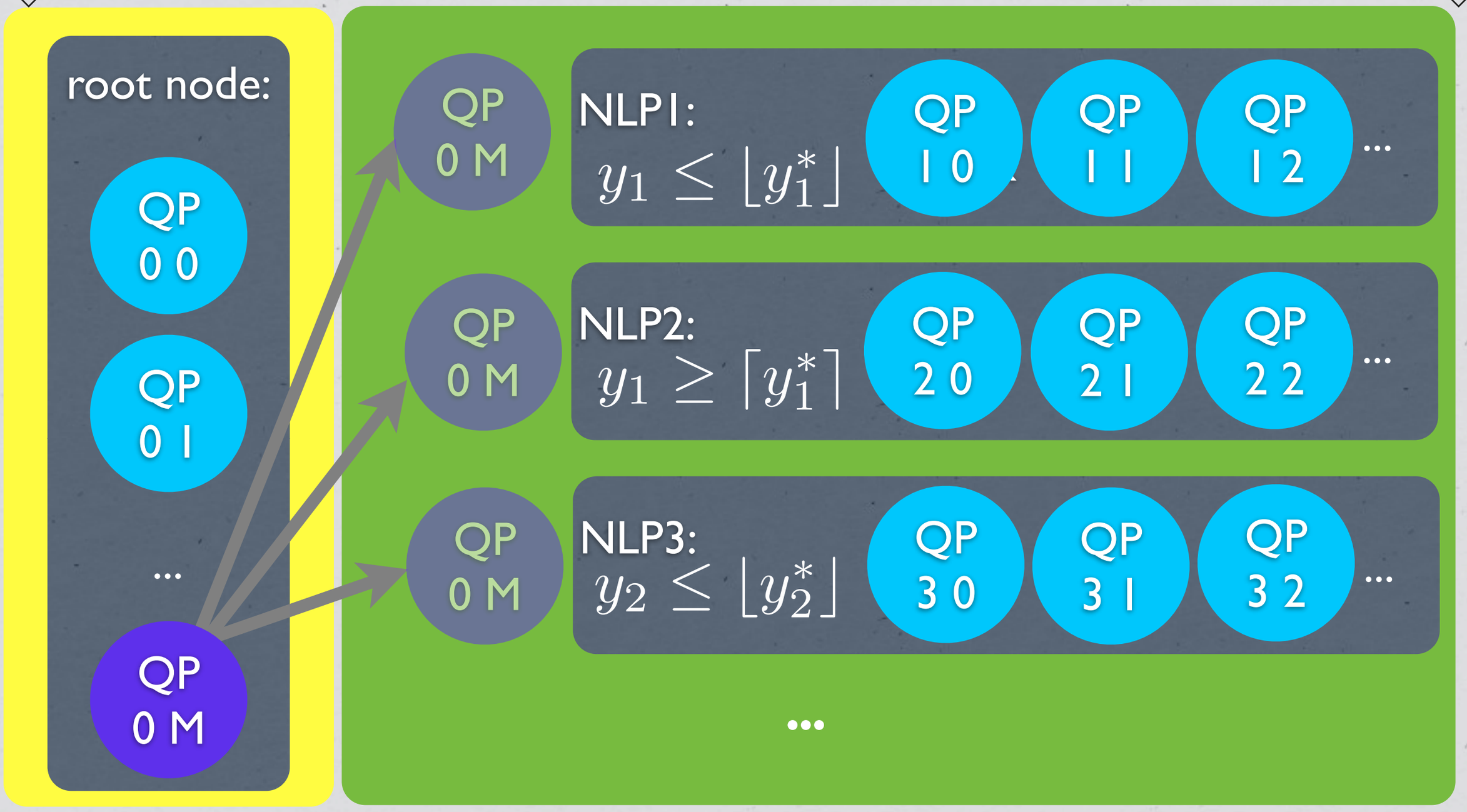
* GOAL: Make good branching decisions

* How:
  * Iteratively fix bounds either up or down from fraction

* Pick variable with largest impact to branch on

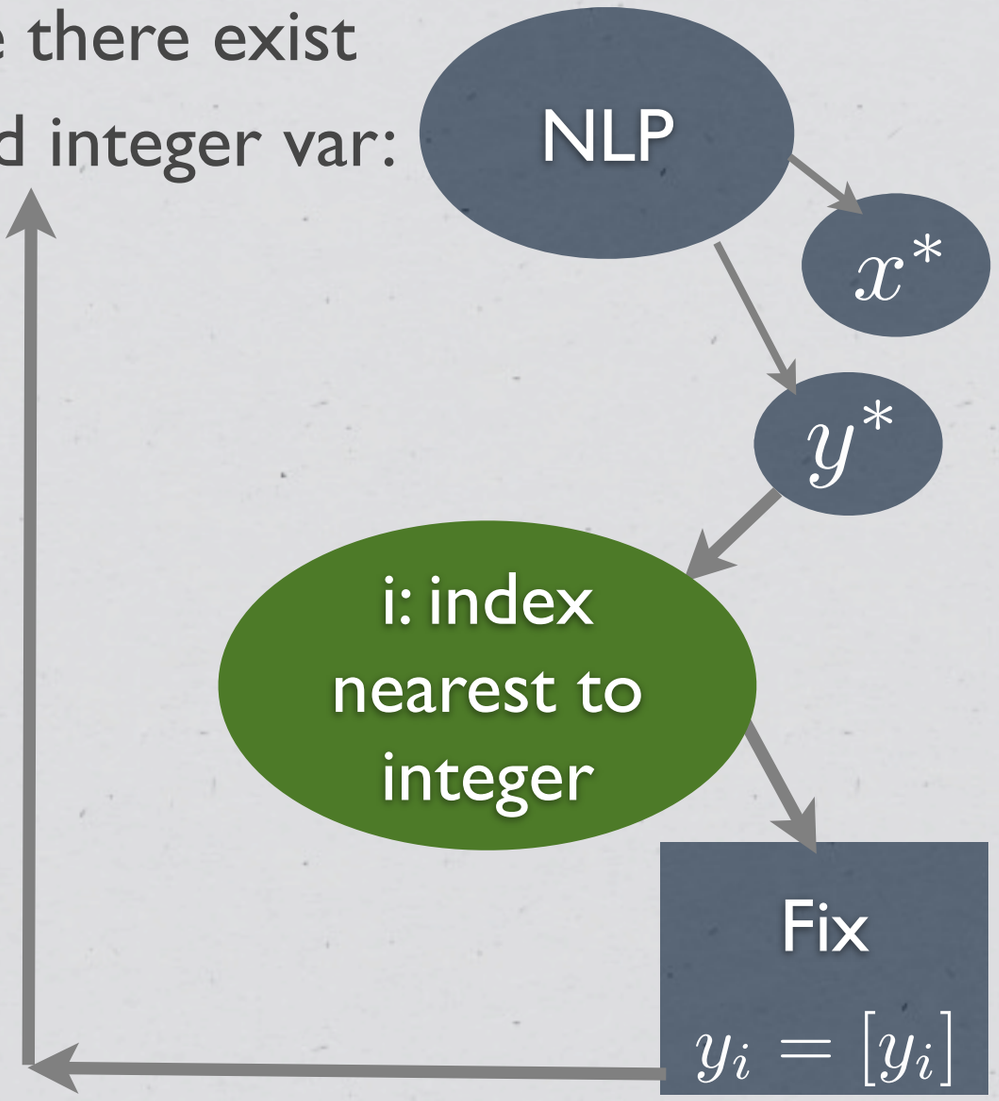


MINLP Strong Branching

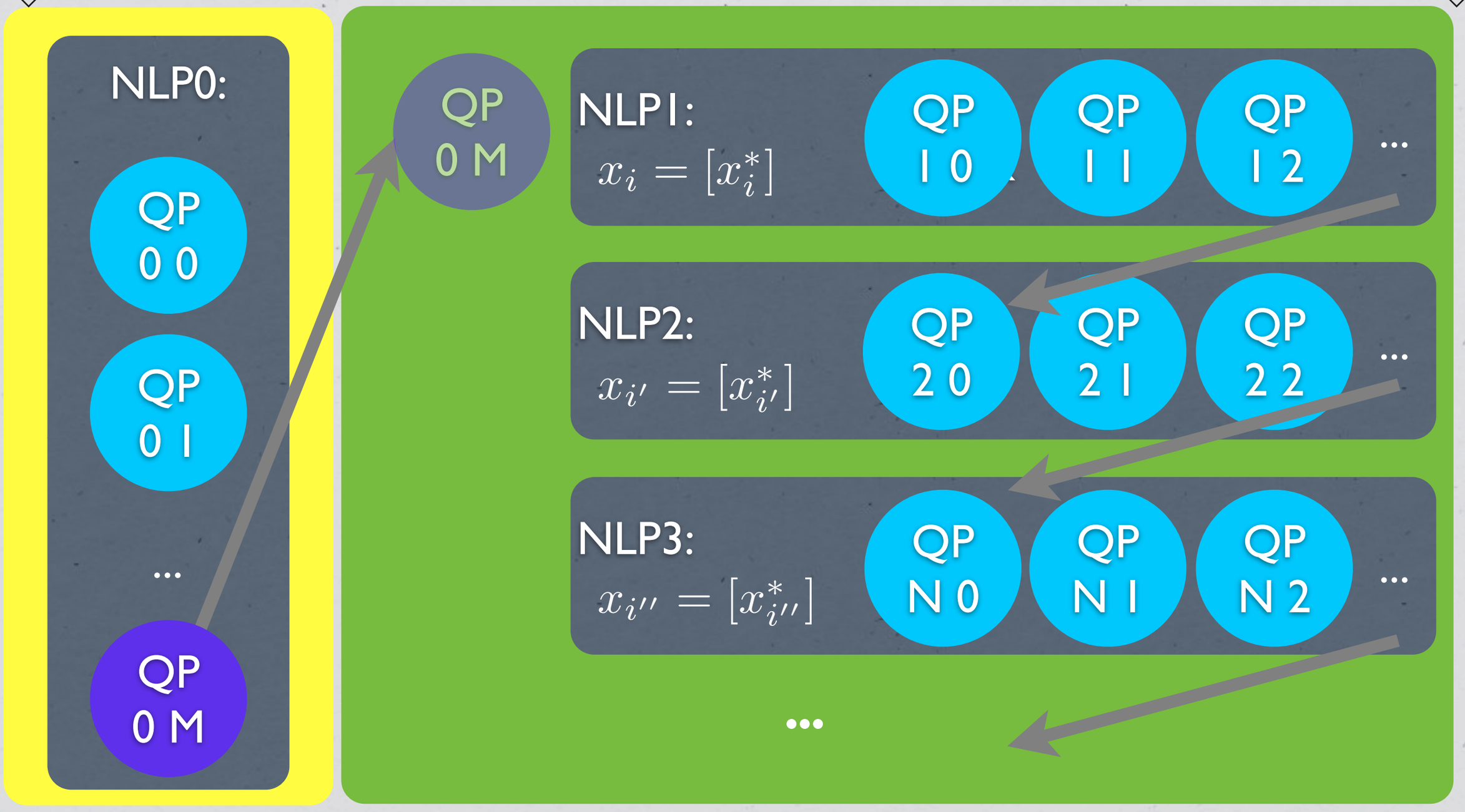


MINLP Diving Heuristic

* GOAL: get an integer feasible solution as incumbent



MINLP Diving Heuristic



Introducing iQP

$$\begin{aligned} \min_{x,y} \quad & \frac{1}{2} d^T W d + g^T d \\ \text{s.t.} \quad & Ad + c = 0 \\ & d \geq l \end{aligned}$$

* GOAL: Iteratively Calculate d using QP hotstarts:
* Iterative Refinement Mode
* SQMR Mode

iQP, diagrammatically

for i=0, 1,...

Inequality Constraints and Iterative Refinement

$$\min_p \ \frac{1}{2} p^T W_0 p + \left(g + W d_i + A^T \lambda_i^+\right)^T p$$

$$\text{s.t.} \ \ A_0 p + (c + A d_i) = 0 \qquad \longrightarrow p_i^\lambda$$

$$p + d_i \geq l$$

Update $(d_{i+1},\ \lambda_{i+1})$

$$d_{i+1} = d_i + p$$

Update $(d_{best},\ \lambda_{best})$

iQP, diagrammatically

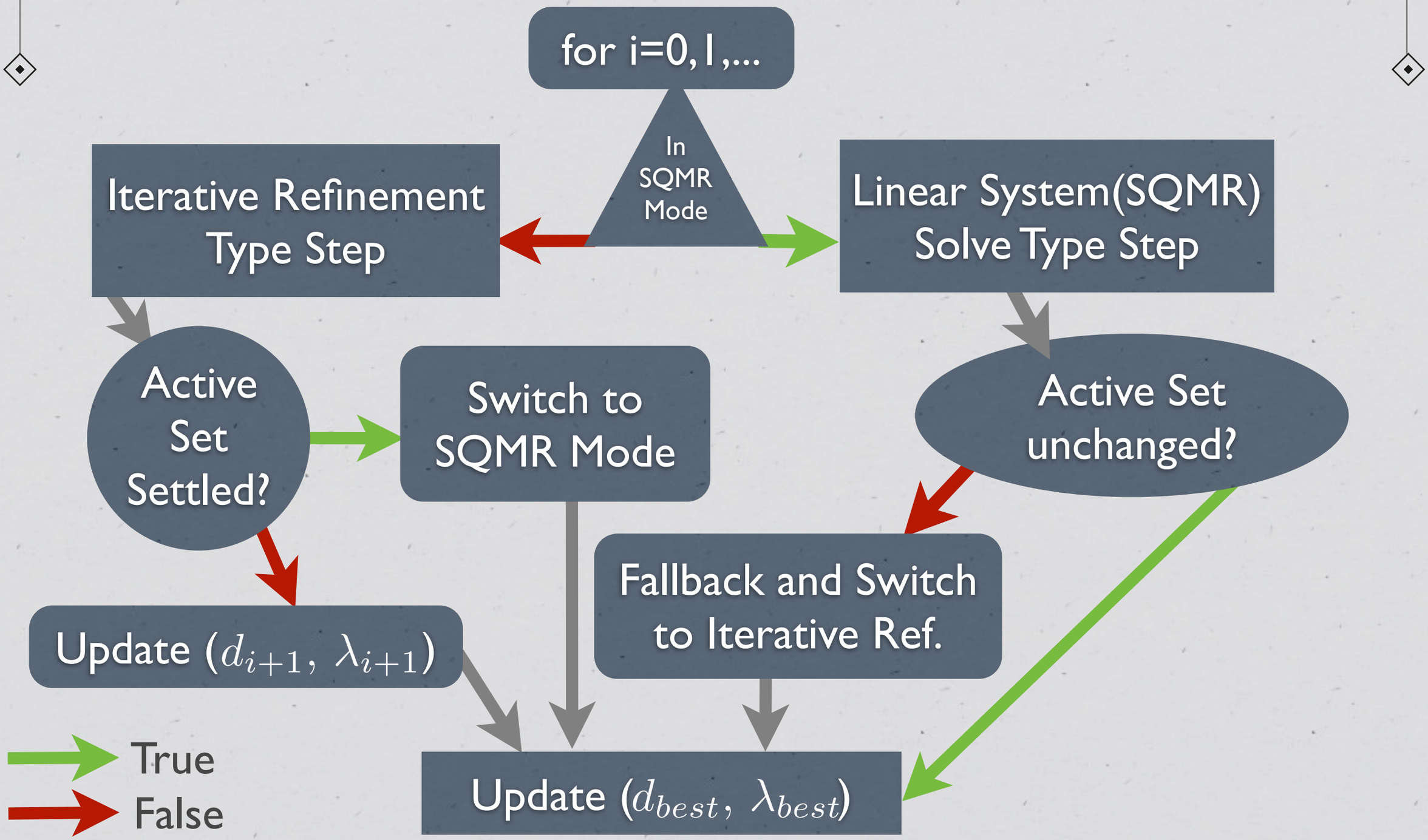
for i=0, 1,...

Preconditioner QP

$$\begin{aligned} \min_z \ & \frac{1}{2} z^T W_0 z + (-r_g^{\mathcal{F}})^T z^{\mathcal{F}} + \left(g^{\mathcal{A}} + W^{\mathcal{A},\cdot} d_i + (A^{\mathcal{A}})^T \lambda_i^+ \right)^T z^{\mathcal{A}} \\ \text{s.t.} \ & A_0 z - r_c = 0 \\ & z^{(j)} \geq 0 \text{ for } j \in \mathcal{A} \end{aligned}$$

Update $(d_{i+1},\ \lambda_{i+1})$

iQP, diagrammatically



QP Solver: qpOASES

* **qpOASES**[Ferreau et al., 2008, Potschka et al., 2012)
  * Open source **parametric** QP solver in C++
  * Added sparse Schur-complement approach [Gill et al., 1990]
* Sparse vs dense *factorizations*--impacts bottleneck!
* Tolerance for QPs: ϵ_{QP}= 1e-8

First Pass: Random ℓ_1QP

* n "d" variables, $m = \frac{1}{2}n$ equality constraints
* W_0, A_0 generated with 5 nonzero/col with random value $O(1)$
* Average over 25 runs per size and perturbation p

QP Data	Perturbation
W, A	$O(p)$
g, c, d_L, d_U	$O(0.1)$

$$
\begin{aligned}
\min \quad & \frac{1}{2} d^T W d + g^T d + 1^T (p + n) \\
\text{s.t.} \quad & Ad + c - p + n = 0 \\
& d_L \leq d \leq d_U \\
& p, n \geq 0
\end{aligned}
$$

Random QP Results (Dense Linear Algebra)

		iQP				warm starts		
				iQP	qpOASES			qpOASES
size	pert	c	CPU	It	(hot) it	c	CPU	it
100	**0.01**	25	4.4e-3	7.92	19.00	25	57.52e-3	8.00
100	**0.025**	25	6.4e-3	10.01	23.52	25	57.75e-3	8.16
100	**0.1**	23	12.5e-3	18.57	50.52	25	71.75e-3	14.20
100	**0.2**	12	30.5e-3	35.58	122.83	25	69.78e-3	23.40
500	**0.01**	25	0.244	8.20	46.88	25	7.20	31.48
500	**0.025**	25	0.358	10.96	59.72	25	6.81	33.56
500	**0.1**	25	1.063	26.20	167.08	25	6.83	64.04
500	**0.2**	0	--	--	--	25	6.83	107.92
1000	**0.01**	25	3.36	8.60	84.64	25	67.58	62.36
1000	**0.025**	25	4.05	11.04	105.80	25	68.29	67.92
1000	**0.1**	15	11.23	27.00	281.13	25	71.16	123.36
1000	**0.2**	1	24.60	52.00	648.00	25	74.70	209.60

Random QP Results (Sparse Linear Algebra)

		iQP				warm starts		
				iQP	qpOASES			qpOASES
size	pert	c	CPU	lt	(hot) it	c	CPU	it
100	**0.01**	25	4.9e-3	7.88	18.76	25	4.87e-3	8.00
100	**0.025**	25	6.4e-3	10.00	23.20	25	5.24e-3	8.16
100	**0.1**	23	16.8e-3	18.57	50.48	25	9.54e-3	14.24
100	**0.2**	12	47.2e-3	35.58	123.50	25	18.82e-3	23.44
500	**0.01**	25	0.079	8.16	46.8	25	0.150	31.48
500	**0.025**	25	0.118	10.96	59.76	25	0.165	33.56
500	**0.1**	25	0.382	26.12	166.40	25	0.309	64.04
500	**0.2**	0	--	--	--	25	0.625	107.92
1000	**0.01**	25	0.439	8.60	84.60	25	0.517	62.32
1000	**0.025**	25	0.530	11.04	105.76	25	0.406	67.92
1000	**0.1**	15	1.44	27.00	280.93	25	0.774	123.36
1000	**0.2**	1	3.30	52.00	649.00	25	1.319	209.60

Numerical Framework: P-SQP

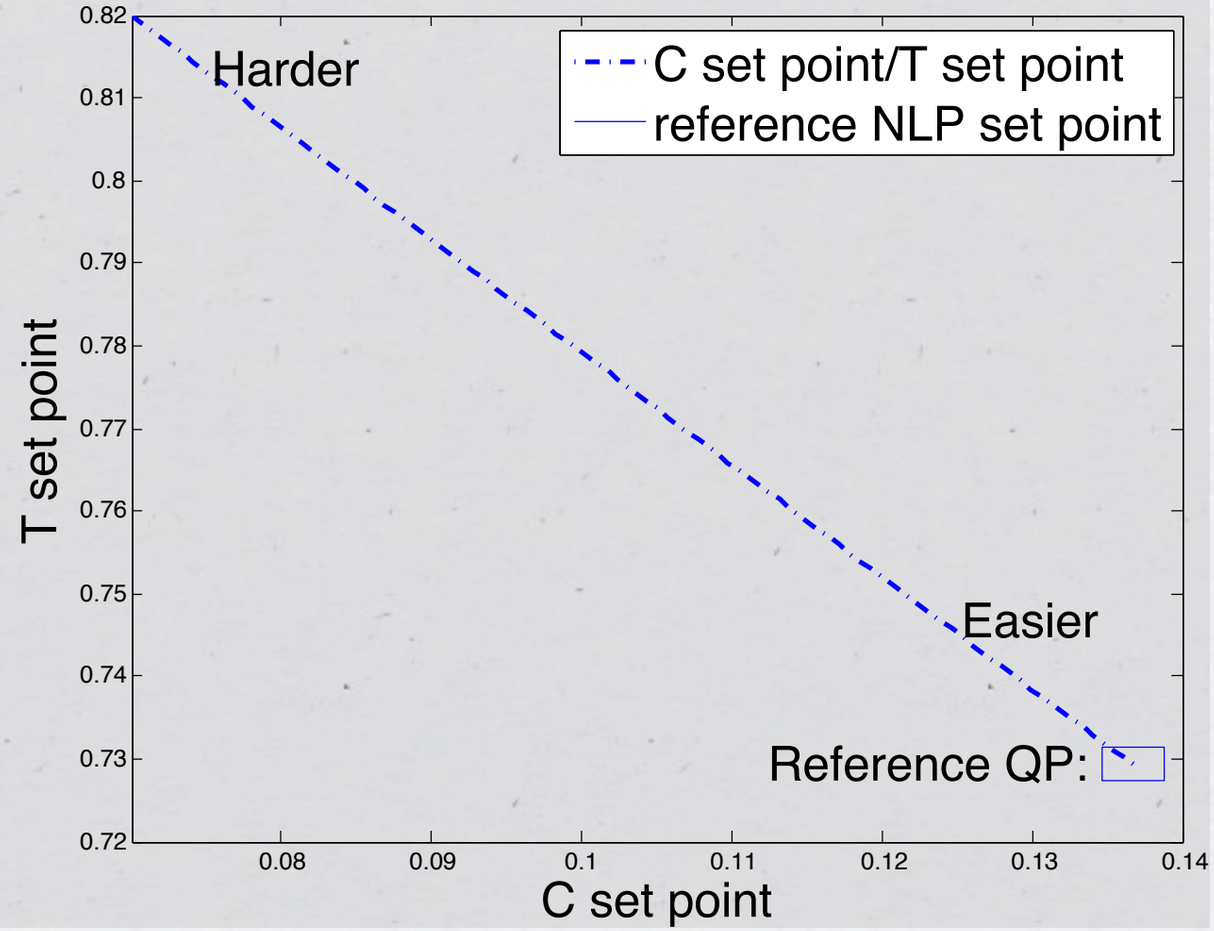
* Used P-SQP [Curtis]: SQP with penalty parameter in quadratic model and $\mathrm{S}\ell_1\mathrm{QP}$ formulation
  * Penalty: fast infeasibility detection for infeasible MINLP nodes
  * $\mathrm{S}\ell_1\mathrm{QP}$: QP is always feasible
* Tolerance: ϵ_{NLP} = 1e-6

NMPC Experiment: Hicks-Ray Reactor

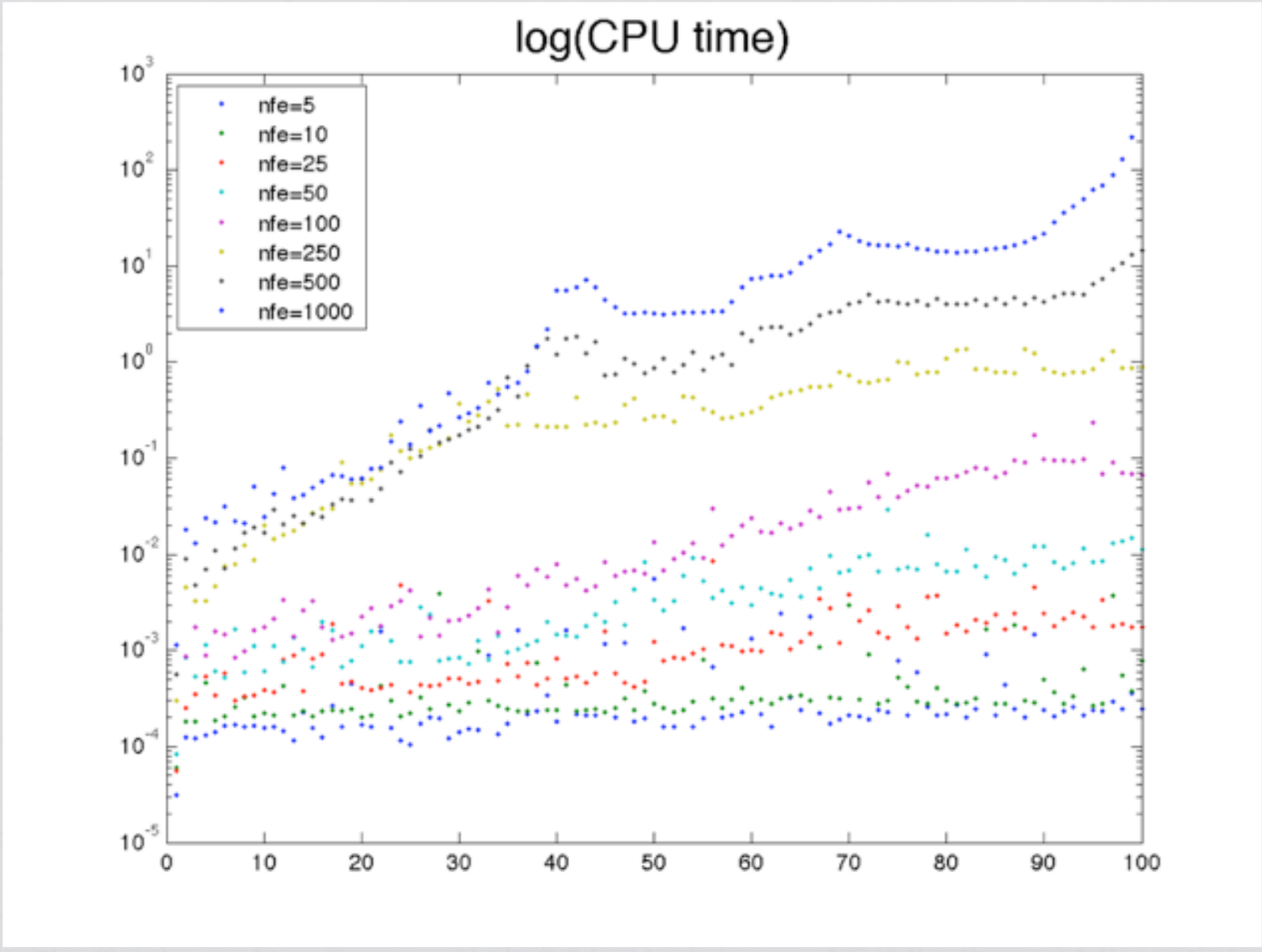
$$\dot{C}_i = (1 - C_i)/\theta - k_{10} \exp(-n/T_{des}) C_i$$

$$\dot{T}_i = (y_f - T_i)/\theta + k_{10} \exp(-n/T_{des}) C_i - \alpha U_i (T_i - y_c)$$

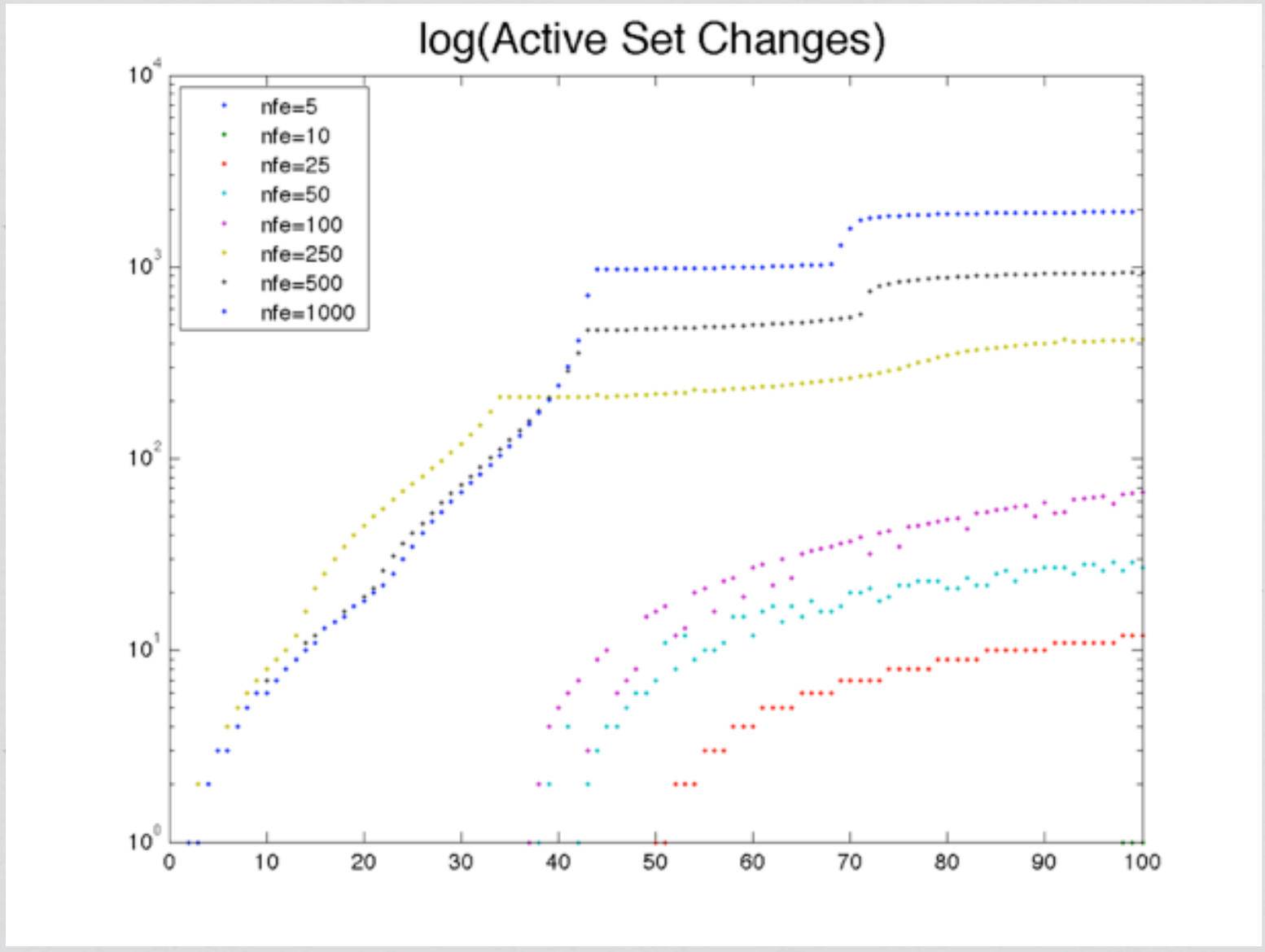
* Linearize Hicks-Ray Reactor about $T_i = T_{des}$
* Vary T_{des}, C_{des}, U_{des}:
  * 100 values along line



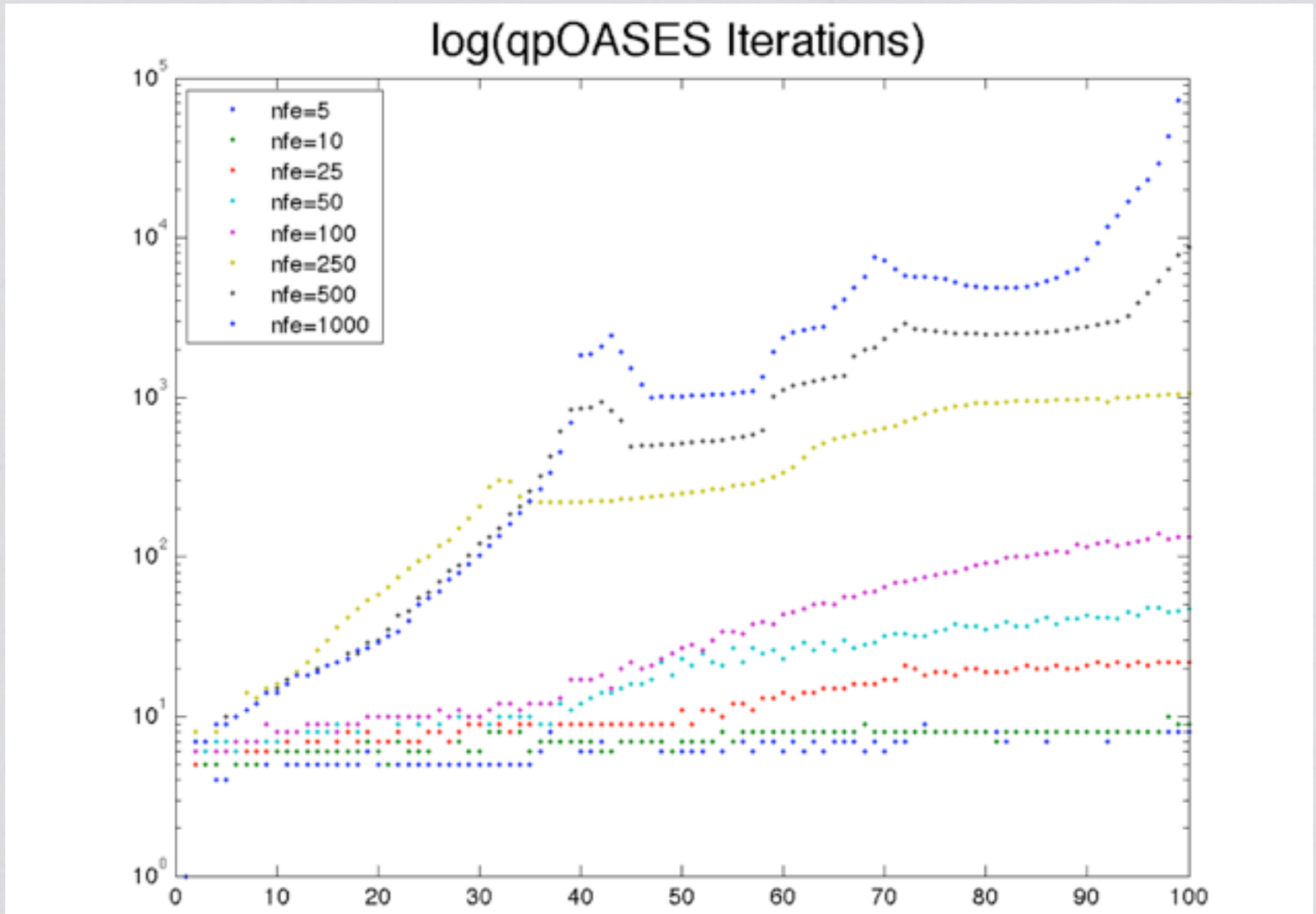
Hicks-Ray CPU Time



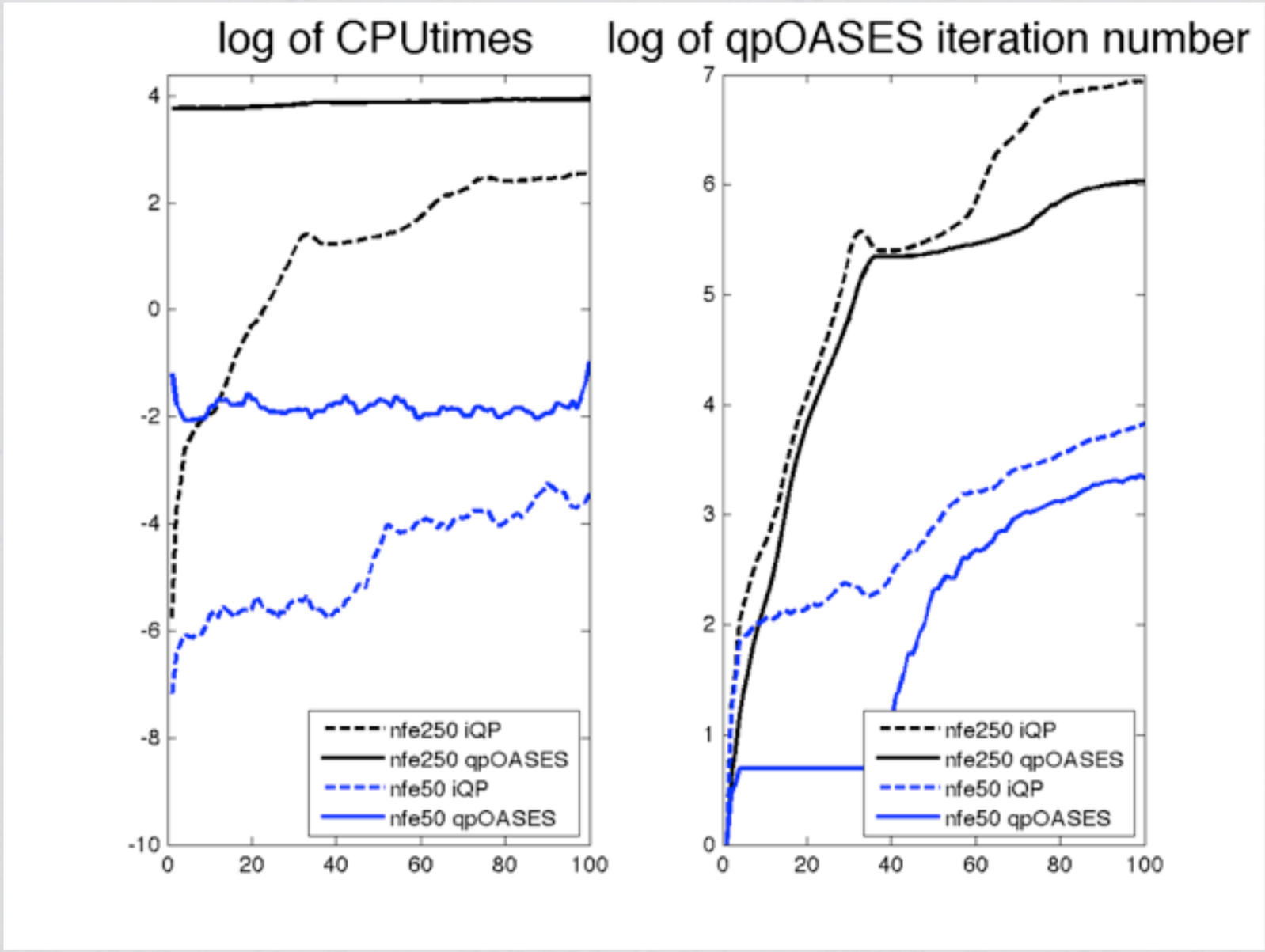
Hicks-Ray: Active Set Changes



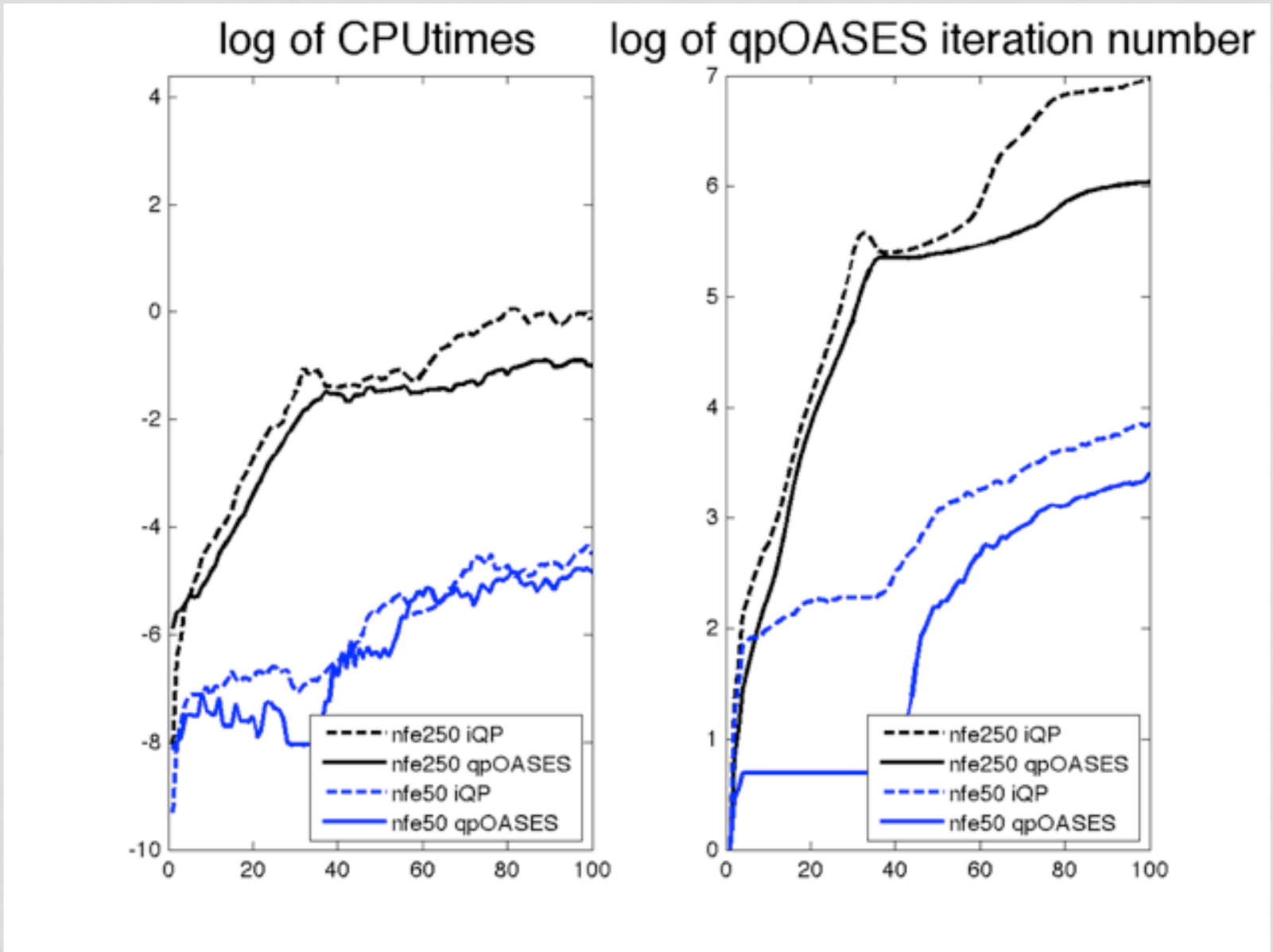
Hicks-Ray: QP Solver Iterations



Hicks-Ray Results: Dense Linear Algebra



Hicks-Ray Results: Sparse Linear Algebra



Uncapacitated Facility Location: congestion and log

$$\min \quad \sum_{j \in \mathcal{C}} z_j + \sum_{i \in \mathcal{F}} p_j y_i$$

$$\text{s.t.} \quad z_j \geq \sum_{i \in \mathcal{F}, j \in \mathcal{C}} a_{ij} COST(x_{ij}) \qquad \forall j \in \mathcal{C}$$

$$x_{ij} \leq z_i \qquad \forall i \in \mathcal{F}, j \in \mathcal{C}$$

$$\sum_{i \in \mathcal{F}, j \in \mathcal{C}} x_{ij} = 1$$

$$0 \leq x_{ij} \leq XUP \qquad \forall i \in \mathcal{F}, j \in \mathcal{C}$$

$$y_j \in \{0, 1\}$$

	COST	XUP
Congest	$1/(1 - x_{ij}/c_{ij})$	$c_{ij} - \epsilon$
LOG	$-\log(1 + x_{ij})$	∞

Strong Branching - Dense

Congestion

			iQP				warm starts		
n	#NLP	SQP/ NLP	NLP failure	CPU[a]	iQP iter[a]	qpOASE iter/qp[a]	NLP failure	CPU[a]	qpOASES iter/qp[a]
95	10	38.1	0	2.53e-	8.74	17.85	0	40.7e-2	2.52
132	12	34.0	4	8.95e-	10.36	32.08	0	117.9e-2	5.24
175	14	35.1	2	31.82e	10.22	43.66	0	351.2e-2	3.42
224	16	34.5	3	123.22	8.53	60.80	0	839.5e-2	3.91

Log Costs

			iQP				warm starts		
n	#NLP	SQP/ NLP	NLP failure	CPU[a]	iQP iter[a]	qpOASE iter/qp[a]	NLP failure	CPU[a]	qpOASES iter/qp[a]
95	10	29.3	0	1.20e-	8.59	19.59	0	10.4e-2	1.51
132	11	34.0	1	2.80e-	9.66	27.50	0	27.5e-2	1.75
175	14	29.5	1	5.24e-	8.95	29.37	0	73.0e-2	1.65
224	16	32.4	1	22.48e	9.79	50.69	0	150.8e-2	1.66

Strong Branching - Sparse

Congestion

			iQP				warm starts		
n	#NLP	SQP/ NLP	NLP failure	CPU[a]	iQP iter[a]	qpOASES iter/qp[a]	NLP failure	CPU[a]	qpOASES iter/qp[a]
95	10	37.8	0	8.62e-	8.74	18.89	0	1.48e-3	2.52
340	20	31.4	5	19.75e	8.87	20.87	0	3.55e-3	2.00
735	30	29.5	11	55.88e	7.85	29.63	0	8.68e-3	2.05
1280	36	17.61	9	192.85	8.89	58.40	0	71.09e-3	19.66

Log Costs

			iQP				warm starts		
n	#NLP	SQP/ NLP	NLP failure	CPU[a]	iQP iter[a]	qpOASE iter/qp[a]	NLP failure	CPU[a]	qpOASES iter/qp[a]
95	10	28.9	2	2.37e-	6.57	10.40	0	0.89e-3	15.66
340	17	20.0	0	8.52e-	7.41	13.01	0	2.18e-3	2.42
735	21	33.1	9	63.94e	9.77	51.02	0	4.42e-3	2.65
1280	28	31.5	6	68.83e	8.18	35.45	0	8.80e-3	3.01

Diving Heuristic

Log: Dense

			iQP				warm starts		
n	#NLP	SQP/ NLP	NLP failure	CPU[a]	iQP iter[a]	qpOAS iter/qp[a]	NLP failure	CPU[a]	qpOAS iter/qp[a]
95	5	25.4	0	7.66e-3	10.55	16.51	0	1.06e-1	1.43
132	6	14.5	0	7.86e-3	8.43	12.42	0	2.71e-1	1.62
175	7	10.6	0	1.45e-2	6.68	11.19	0	7.03e-1	2.12
224	8	19.6	0	2.71e-2	8.18	12.82	0	1.44e0	1.71

Log Sparse

			iQP				warm starts		
n	#NLP	SQP/ NLP	NLP failure	CPU[a]	iQP iter[a]	qpOAS iter/qp[a]	NLP failure	CPU[a]	qpOAS iter/qp[a]
95	5	17.4	0	2.72e-3	8.51	12.91	0	5.88e-4	1.58
735	15	12.2	0	1.68e-2	10.91	20.68	0	6.76e-3	5.39
1280	20	10.1	0	5.53e-2	12.81	36.31	0	1.81e-2	10.00
2820	30	9.23	0	9.09e-2	12.78	32.36	0	2.96e-2	8.55

Outlook

* Good news:
  * Converges in practice
  * good performance when factorization time dominates
* Bad news:
  * slow for realistic problems so far

Limitations

* No globalization scheme…
  * but maybe fix out-of-band
* qpOASES sparse matrix-vector multiplication
* still looking for large dense problems or hard to factor network problems.

Conclusion

* Good method when factorizations are expensive relative to backsubstitutions

* Thank you!