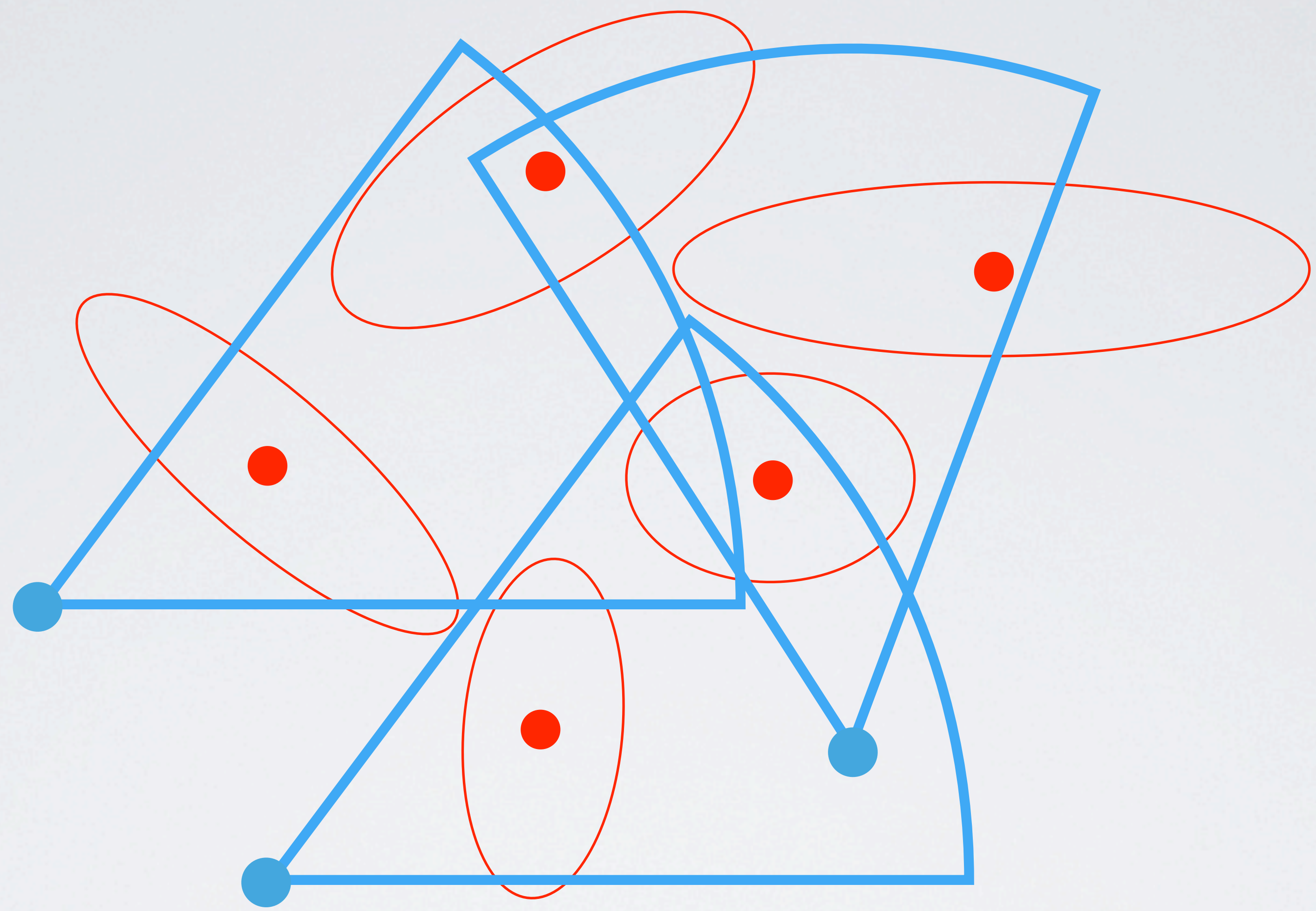
Maximizing Information Gain in Directional Sensor Problems

Hans D. Mittelmann
Arizona State University

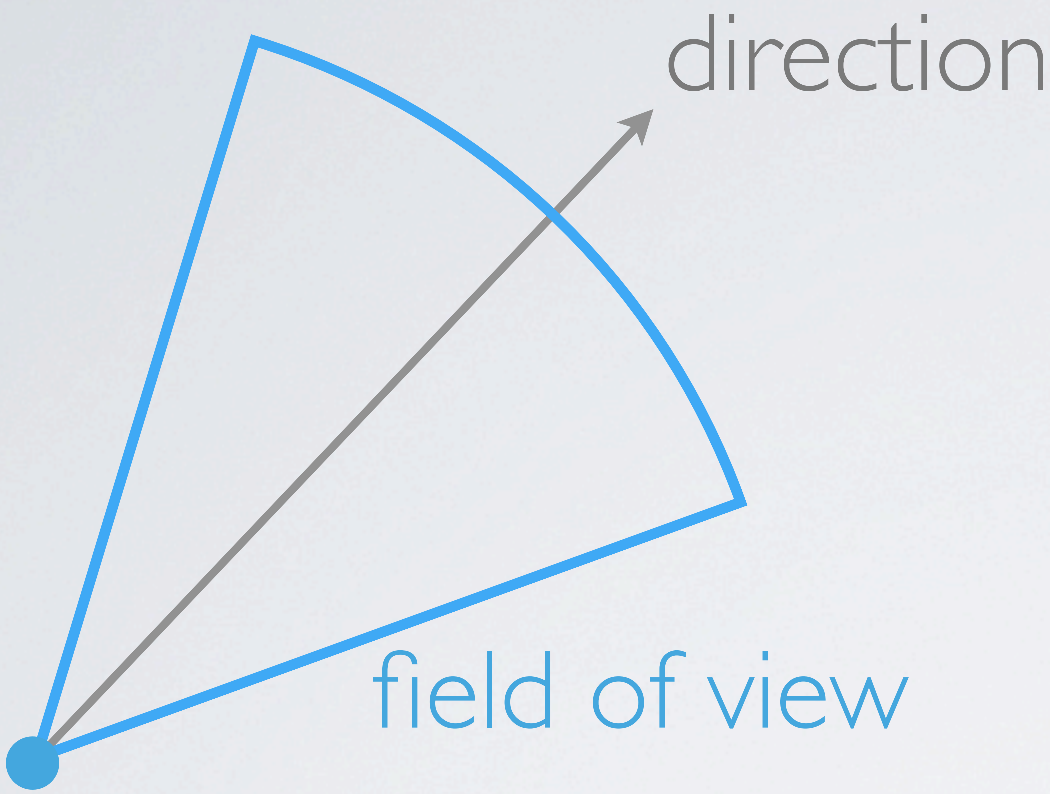
Domenico Salvagnin*
DEI, University of Padova

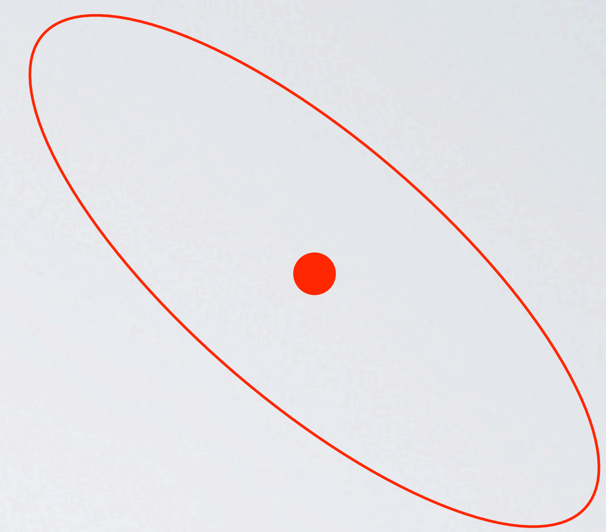
1. Problem



N targets on a 2D plane (with *uncertain* location)

M directional sensors on a 2D plane (with *known* location)





- K discrete values for sensor directions
- control vector **u** associates to each sensor a variable u_i encoding the sensor direction

- target j with location χ_j
- a priori distribution of target location is Gaussian

$$\mathcal{N}(a_j, A_j)$$

- If target j is within the field of view of sensor i, we get the noisy measurement z_{ij} (nothing otherwise):

$$z_{ij} = H\chi_j + \eta_{ij}$$

measurement covariance matrix R

observation model

target location

$$\eta_{ij} \sim \mathcal{N}(0, R(s_i, u_i, \chi_j))$$

- All observations are fused together and approximated as an a posteriori Gaussian distribution $\mathcal{N}(y_j, P_j)$

$$P_j = \left(A_j^{-1} + \sum_{i:visible} H^T (R(s_i, u_i, a_j))^{-1} H \right)^{-1}$$

$$y_j = P_j \left(A_j a_j + \sum_{i:visible} H^T (R(s_i, u_i, a_j))^{-1} z_{ij} \right)$$

Which objective function?

- previous approaches dealt with very combinatorial objective functions
  - maximize coverage
  - minimize number of sensors needed
  - etc...
- Here we **maximize** the total **information gain**
- Note that some combinatorial objectives can be seen as proxies to our objective.

- Given a control vector u, for a given target j the information gain reads:

$$I_j(u) = -\log\left(\frac{\det(P_j)}{\det(A_j)}\right)$$

- The overall objective is thus:

$$\max E\left[\sum_{j=1}^{N} I_j(u)\right]$$

(can be approximated with a Monte Carlo approach)

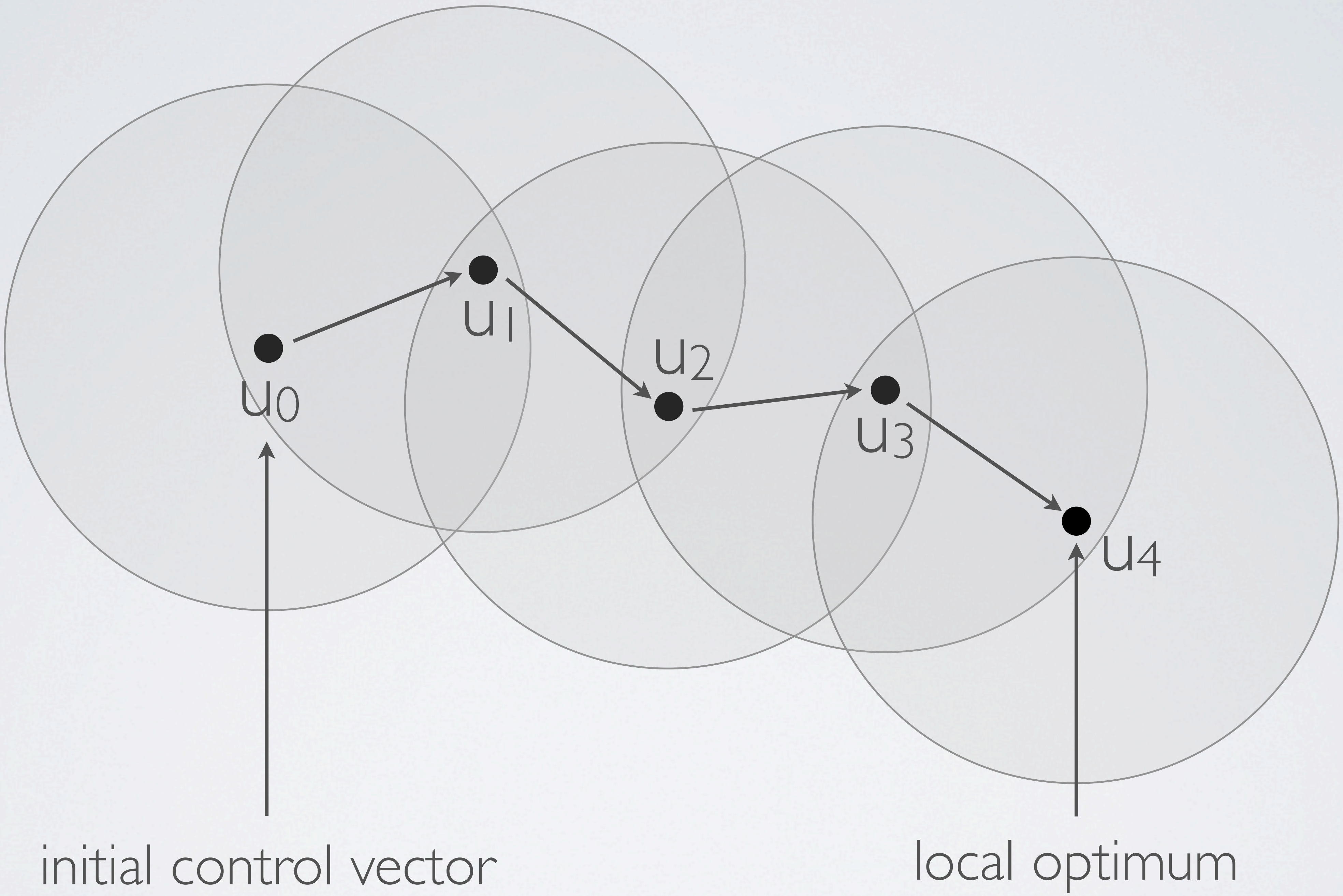
Related Work

- In [1] the information gain objective was introduced
- A few simple ad-hoc heuristics are proposed and a nonlinear programming problem is solved to provide upper bounds
- Heuristics provide reasonably good solutions in polynomial time
- MATLAB code, hard to judge performance

[1] S. Ravi, E. Chong and H. D. Mittelmann, *Cooperative Control of Directional Sensors to Maximize Information Gain*, Proceedings of SPIE conference "Signal and Data Processing of Small Targets 2013", San Diego, CA

2. *Heuristic Methods*

Local Search



Neighborhood defined as the control vectors that can be obtained by changing a single sensor at the time

Meta Heuristics

- Meta-heuristics built on top of local search:
  - **Tabu Search**
  - **Randomized Local Search**
- Both are started from a random control vector
- Both are stopped if no improvement for a given number of iterations

3. *Exact Methods*

Properties

- By algebraic manipulation, it is possible to get rid of inversion of matrices of variables.
- It is possible to compute off-line if a given target j in sample s is visible from sensor i pointing in direction k.
- Most nonlinear expressions can be computed off-line as well.
- log(det(X)) is a concave function in the semidefinite cone.

problem can be formulated as a mixed-integer convex program!

maximize average information gain over all samples

$$\max \sum_s \sum_j \left[\log(\det(\overline{P}_{js})) + \log(\det(A_j))\right] / |S|$$

$$\sum_k u_{ik} = 1 \quad \forall i$$

each sensor must point in one direction

$$\overline{P}_{js} = A_j^{-1} + \sum_i \sum_k R_{ijks} u_{ik} \quad \forall j \forall s$$

definition of inverse of a posteriori covariance matrix (for each target in each sample)

↑ inverse of a posteriori covariance matrix

↖ inverse of measurement covariance matrix (or null matrix if not visible)

How to solve it?

Can be modeled easily with an algebraic modeling language such as AMPL and fed directly to a Mixed-Integer Convex Programming solver, such as SCIP or KNITRO

- easy to implement
- MICP solvers are not however as stable as MIP solvers...
- finding the right solver/parameter tuning can be tricky

Benders!

Devise a generalized Benders decomposition approach and use a Mixed-Integer Programming solver, such as CPLEX

- MIP solvers are a mature and stable technology
- Master problem has only variables u_{ik}, while we have a slave for each target j and each sample s
- Slaves can be solved analytically in our case
- Benders cuts (in this case, outer approximation cuts) can be numerically unstable...

Master

$$\begin{cases} \max \left[\sum_{s,j} \theta_{sj} \right] / |S| \\ \sum_k u_{ik} = 1 \\ \langle \text{Benders cuts} \rangle \\ u_{ik} \in \{0, 1\} \\ \theta_{sj} \quad \text{free} \end{cases}$$

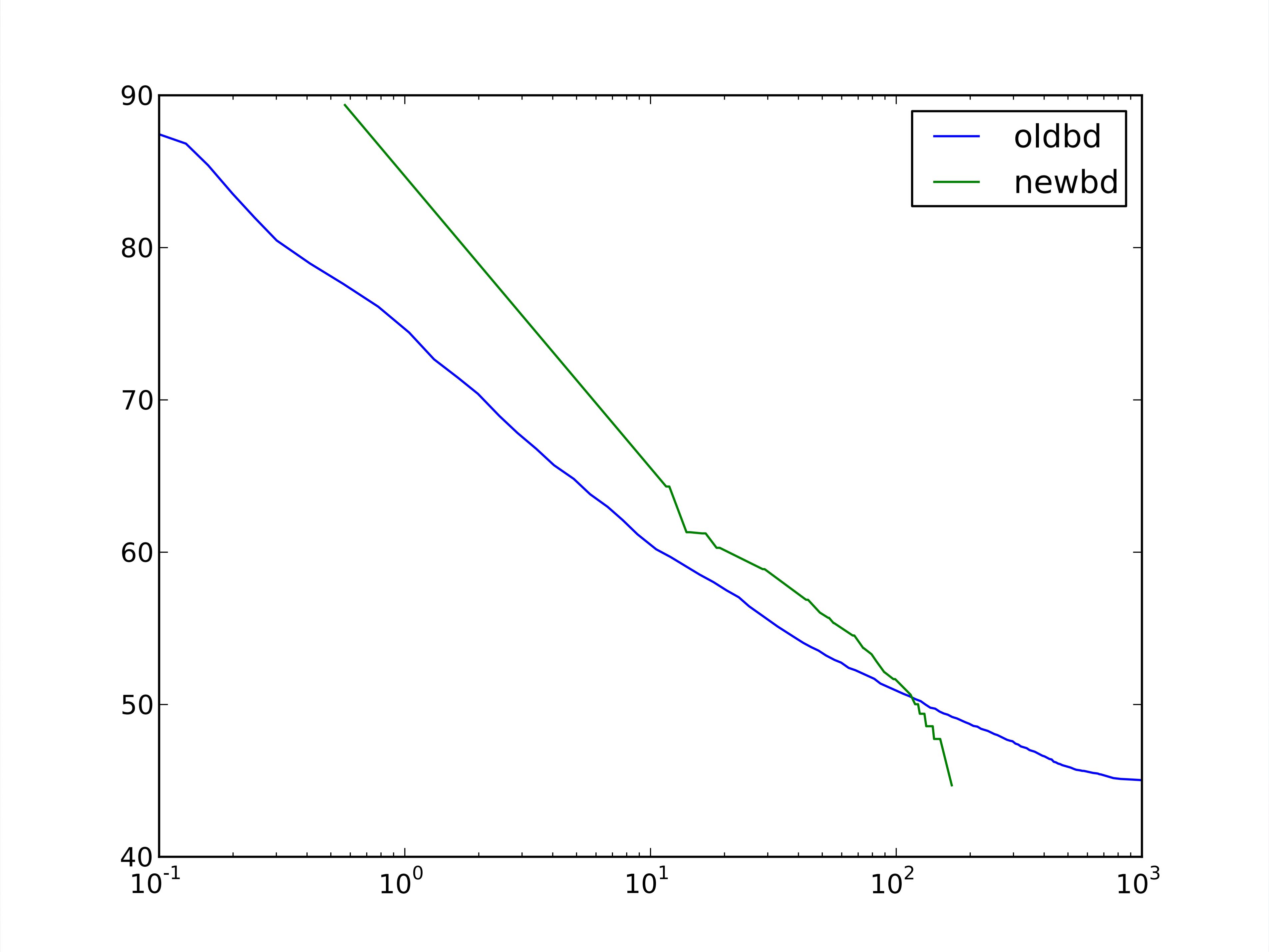
Slaves

$$\begin{cases} f(\overline{P}_{js}) = \log(\det(\overline{P}_{js})) + \log(\det(A_j)) \geq \theta^*_{sj} \\ \overline{P}_{js} = A_j^{-1} + \sum_i \sum_k R_{ijks} u^*_{ik} \end{cases}$$

Benders Cut

$$\theta_{sj} \leq f(\overline{P}^*_{js}) + \nabla f(\overline{P}^*_{js})(\overline{P}_{js} - \overline{P}^*_{js})$$

How to implement Benders?



Old Style Benders

- solve MIP at each iteration
- builtin restarts
- MIP as black box

Modern Benders

- single tree B&C
- dual reductions off
- intrusive callbacks
- bad branching at the very beginning

Hybrid Benders

- best of both worlds
- can do even better when combined with RLS

4. Preliminary Computations

N = 9   K = 10   S = 150

M	TS	RLS	KNITRO	Benders	HBender
4	*0.22	0.79	139.32	22.85	6.12
5	0.38	1.64	409.96	110.20	9.78
6	0.53	2.39	1637.55	398.66	26.03
7	0.68	3.64	9188.71	3660.68	144.38
8	*0.98	4.77	31619.70	20937.65	1757.69

*optimum missed

Running times in seconds

Conclusions

- RLS is slightly more expensive than TS, but always finds the optimal solution (while TS only 3/5).
- KNITRO and a simple Benders implementation do not seem to scale well as the number of sensors increases.
- A more sophisticated Benders implementation performs much better.

Questions?