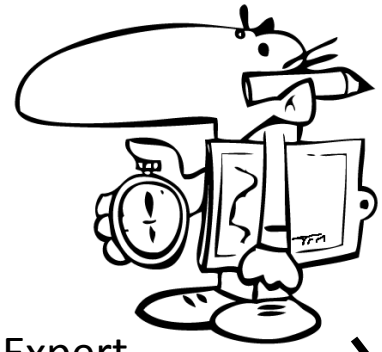
Automatic Solver Configuration and Solver Portfolios

Meinolf Sellmann

IBM Research Watson

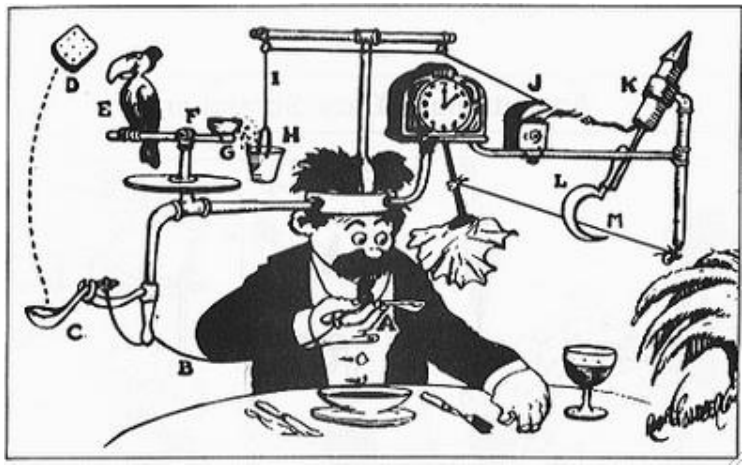
Why Tune Algorithms?



Expert

Develops

Pretunes

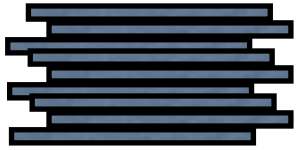


Algorithm

Documents
Parameters

Tune

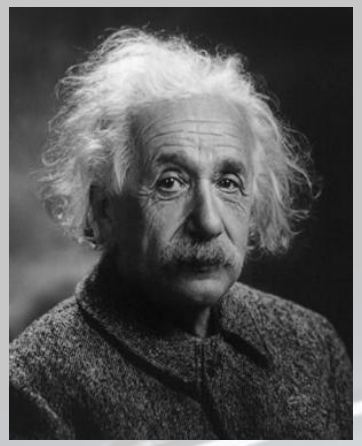
Users

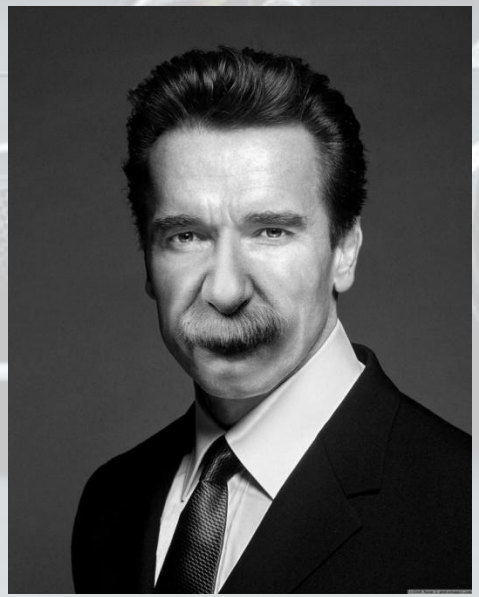


Instances

Why Tune Algorithms?

Tuning vs Configuration





Why Tune Algorithms?

- Algorithms have parameters
  - Implicit in the implementation
  - Open to user
  - Big influence on practical performance (speed, accuracy, robustness, etc)
- The practice: manual tuning
  - Takes a lot of time, often not very good
  - Requires user to learn meaning of parameters
- Objectives
  - Automate tuning
  - Automatic algorithm customization
  - Aid developers in algorithm configuration
  - Enable fair comparison of algorithms

Why Bundle Algorithms?

Why Bundle Algorithms?

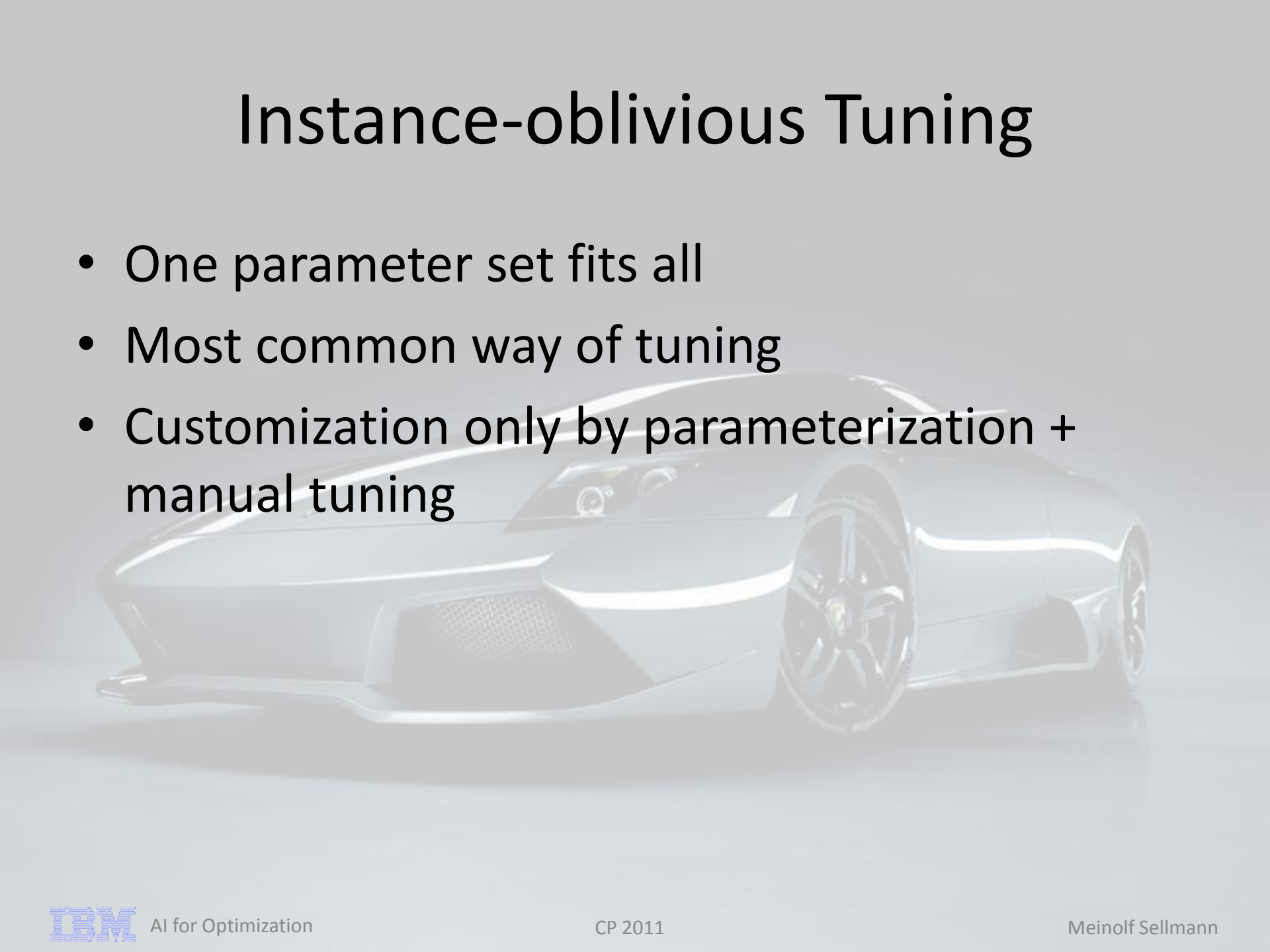
Content

- **Instance-Oblivious Tuning**
  - Overview of Approaches
  - Parameters: Variable Tree Representation
  - GGA: Gender-Based Genetic Algorithm
  - GGA: Numerical Results
- Algorithm Portfolios
  - Overview of Approaches
  - SATzilla
  - CP-Hydra
  - 3S

- Instance-Specific Tuning
  - Overview of Approaches
  - ISAC: Feature-based Parameter Selection
  - ISAC: Numerical Results

Instance-oblivious Tuning

- One parameter set fits all
- Most common way of tuning
- Customization only by parameterization + manual tuning



Overview of Methods

- Popular Tuning Methods
  - Enumerate all configurations
  - Test specific configurations
    (based on some understanding of the parameters)
  - Hand tuning (usually by limited local search)
  - Automated tuning

Overview of Methods

- Continuous parameters
  - Mesh-adaptive Direct Search, MADS [Audet et al, '06]
  - Population-based, e.g. CMA-ES [Hansen et al, '95]
- Categorical parameters
  - Hill-climbing, Composer [Gratch et al, '92]
  - Beam search, MULTI-TAC [Minton, '93]
  - Racing algorithms, F-Race [Birattari et al, '02]
  - CALIBRA [Adenso-Diaz & Laguna, '06]
  - Iterated Local Search, ParamILS [Hutter et al, '07]
- Model-Based Parameter Optimization
  - Sequential Parameter Optimization (SPO) [Bartz-Beielstein et al., '05]
  - Extensions of SPO [Hutter et al, '09]
- Non-model-based configuration for general parameters
  - Gender-based genetic algorithm (GGA) [Ansotegui et al. '09]

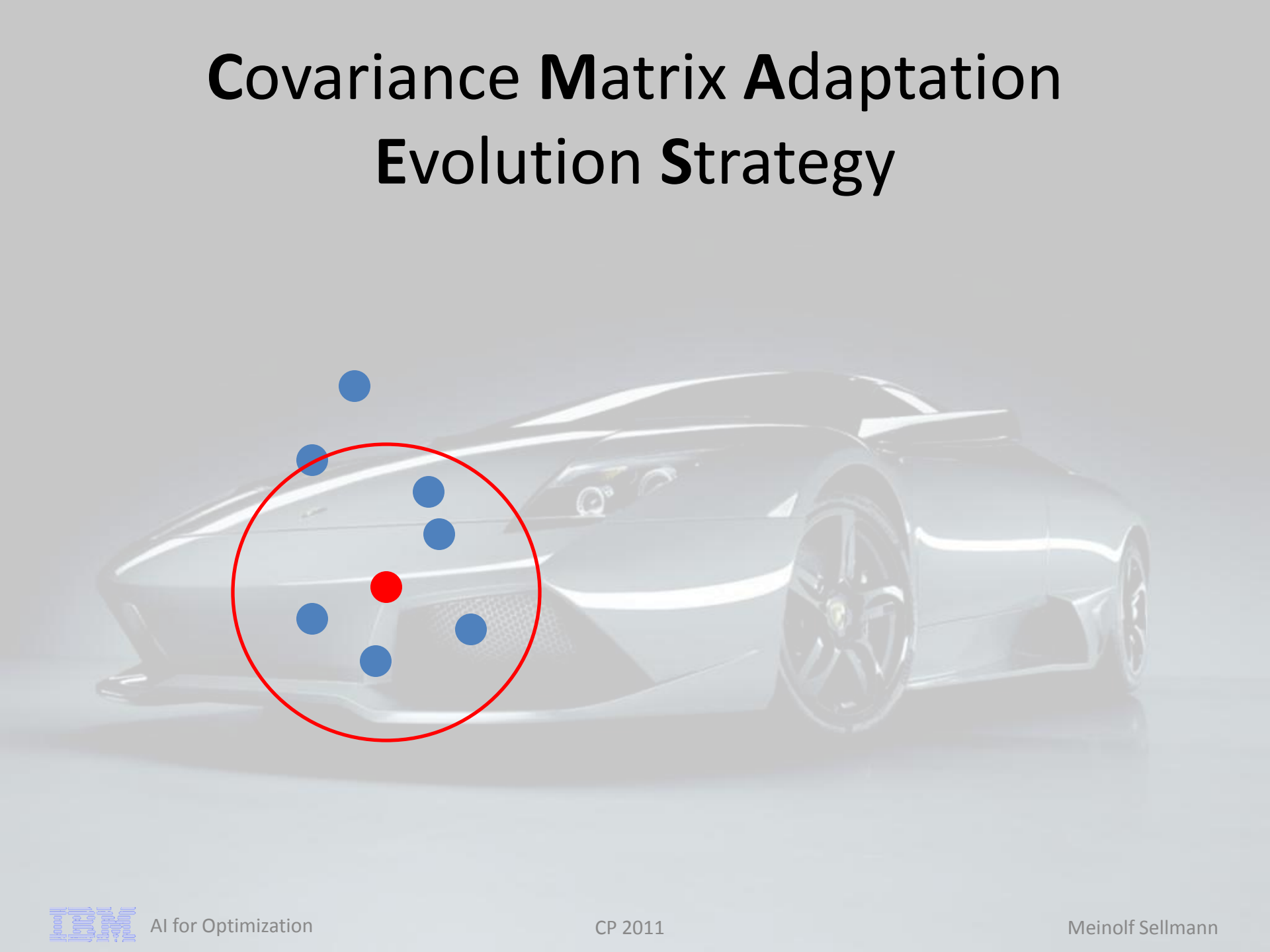
Covariance **M**atrix **A**daptation **E**volution **S**trategy

- General optimizer for highly non-linear continuous optimization problems
- Black box optimization (derivatives not available)
- The typical difference quotients are not useful
- Discontinuities
- Noise and outlier
- Many local optima
- In summary: Black box optimization in a rough or rugged landscape.

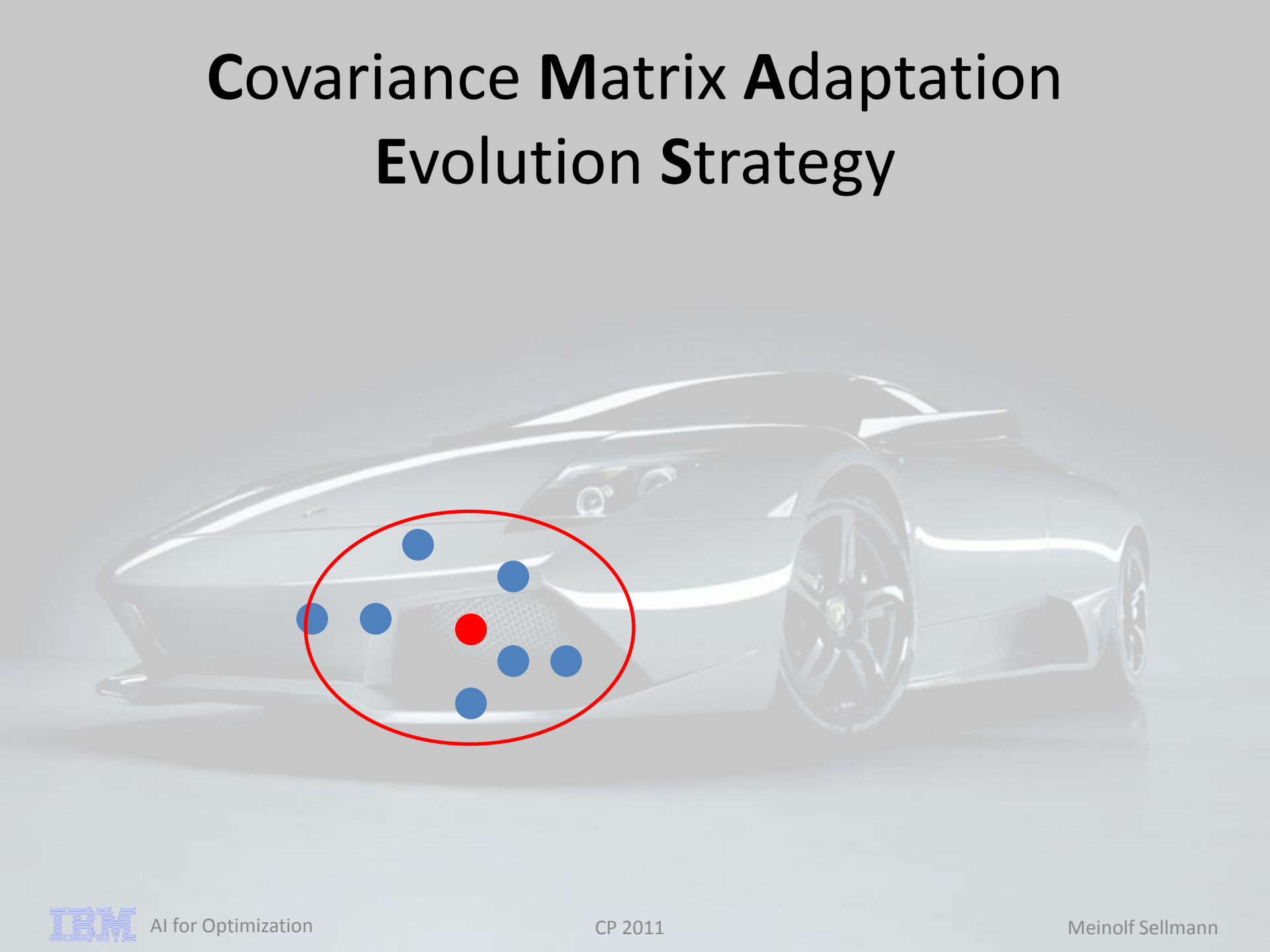
Covariance **M**atrix **A**daptation **E**volution **S**trategy

- Repeat
  - Sample *m* times around "point of interest" □ according to N(□,□).
  - Determine best sampling point and set □ to it.
  - Adapt □.

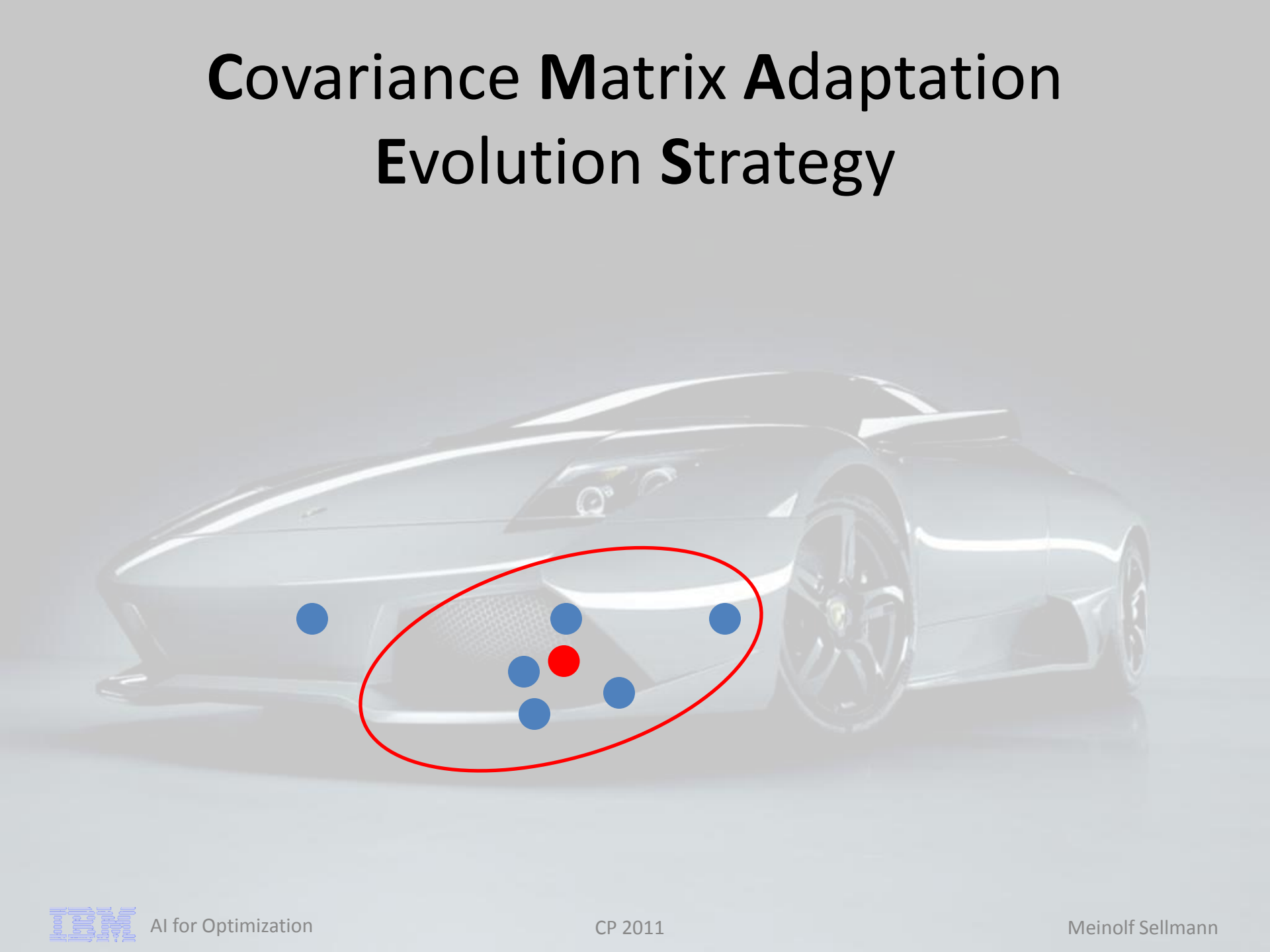
Covariance **M**atrix **A**daptation **E**volution **S**trategy



Covariance **M**atrix **A**daptation **E**volution **S**trategy



Covariance **M**atrix **A**daptation **E**volution **S**trategy



Covariance **M**atrix **A**daptation **E**volution **S**trategy

Consider $P^{(t)} = \mathcal{N}\left(\boldsymbol{\mu}^{(t)},\ {\sigma^{(t)}}^2 \boldsymbol{C}^{(t)}\right)$ where $\boldsymbol{\mu}^{(t)} \in \mathbb{R}^n$, $\sigma^{(t)} \in \mathbb{R}_+$, $\boldsymbol{C}^{(t)} \in \mathbb{R}^{n \times n}$

- $\boldsymbol{\mu}^{(t)} \rightarrow \boldsymbol{\mu}^{(t+1)}$: Maximum likelihood update, i.e. $P(\boldsymbol{x}^{(t)}_{\text{selected}} | \boldsymbol{\mu}^{(t+1)}) \rightarrow \max$
- $\boldsymbol{C}^{(t)} \rightarrow \boldsymbol{C}^{(t+1)}$: Maximum likelihood update, i.e. $P(\frac{\boldsymbol{x}^{(t)}_{\text{selected}} - \boldsymbol{\mu}^{(t)}}{\sigma^{(t)}} | \boldsymbol{C}^{(t+1)}) \rightarrow \max$, under consideration of prior $\boldsymbol{C}^{(t)}$ (otherwise $\boldsymbol{C}^{(t+1)}$ becomes singular).
- $\sigma^{(t)} \rightarrow \sigma^{(t+1)}$: Update to achieve conjugate perpendicularity, i.e. conceptually $(\boldsymbol{\mu}^{(t+2)} - \boldsymbol{\mu}^{(t+1)})^{\mathrm{T}} {\boldsymbol{C}^{(t)}}^{-1} (\boldsymbol{\mu}^{(t+1)} - \boldsymbol{\mu}^{(t)}) / {\sigma^{(t+1)}}^2 \rightarrow 0$

Multi-TAC

- Selector for heuristics in backtrack search
- Beam Search Approach
- Repeat
  - Add a single heuristic to each current search method
  - Evaluate all resulting search methods
  - Keep the best *m* methods

Multi-TAC

(o,o,o)

(1,o,o)

(2,o,o)

(3,o,o)

(4,o,o)

(o,1,o)

(o,2,o)

(o,o,1)

(o,o,2)

(o,o,3)

Multi-TAC

(1,o,o)

(12,o,o) (1,1,o) (1,o,1)

(13,o,o) (1,2,o) (1,o,2)

(14,o,o) (1,o,3)

(3,o,o)

(31,o,o) (3,1,o) (3,o,1)

(32,o,o) (3,2,o) (3,o,2)

(34,o,o) (3,o,3)

Multi-TAC

(34 , o , 2)

(341 , o , 2)

(34 , 1 , 2)

(34, o , 21)

(342, o , 2)

(34 , 2 , 2)

(34 , o , 23)

(3 , o , 2)

(31 , o , 2)

(3 , 1 , 2)

(3 , o , 21)

(32 , o , 2)

(3 , 2 , 2)

(3 , o , 23)

(34 , o , 2)

F-Race

- How to determine whether one meta-heuristic works better than another?
- Repeat
  - Pick a new instance
  - Run and rank all algorithms still in the race
  - Remove inferior algorithms

F-Race

1	2	3	4	5
5	1	6	4	
3	5	3		
2	2	1	2	2
7	6			
6	4	5		
4	7	4	3	
1	3	2	1	1

F-Race

Friedmann Test

$$T = \frac{(n-1)\sum_{j=1}^{n}\left(R_j - \frac{k(n+1)}{2}\right)^2}{\sum_{l=1}^{k}\sum_{j=1}^{n} R_{lj}^2 - \frac{kn(n+1)^2}{4}}.$$

$$\frac{|R_j - R_h|}{\sqrt{\frac{2k\left(1-\frac{T}{k(n-1)}\right)\left(\sum_{l=1}^{k}\sum_{j=1}^{n} R_{lj}^2 - \frac{kn(n+1)^2}{4}\right)}{(k-1)(n-1)}}}$$

GGA

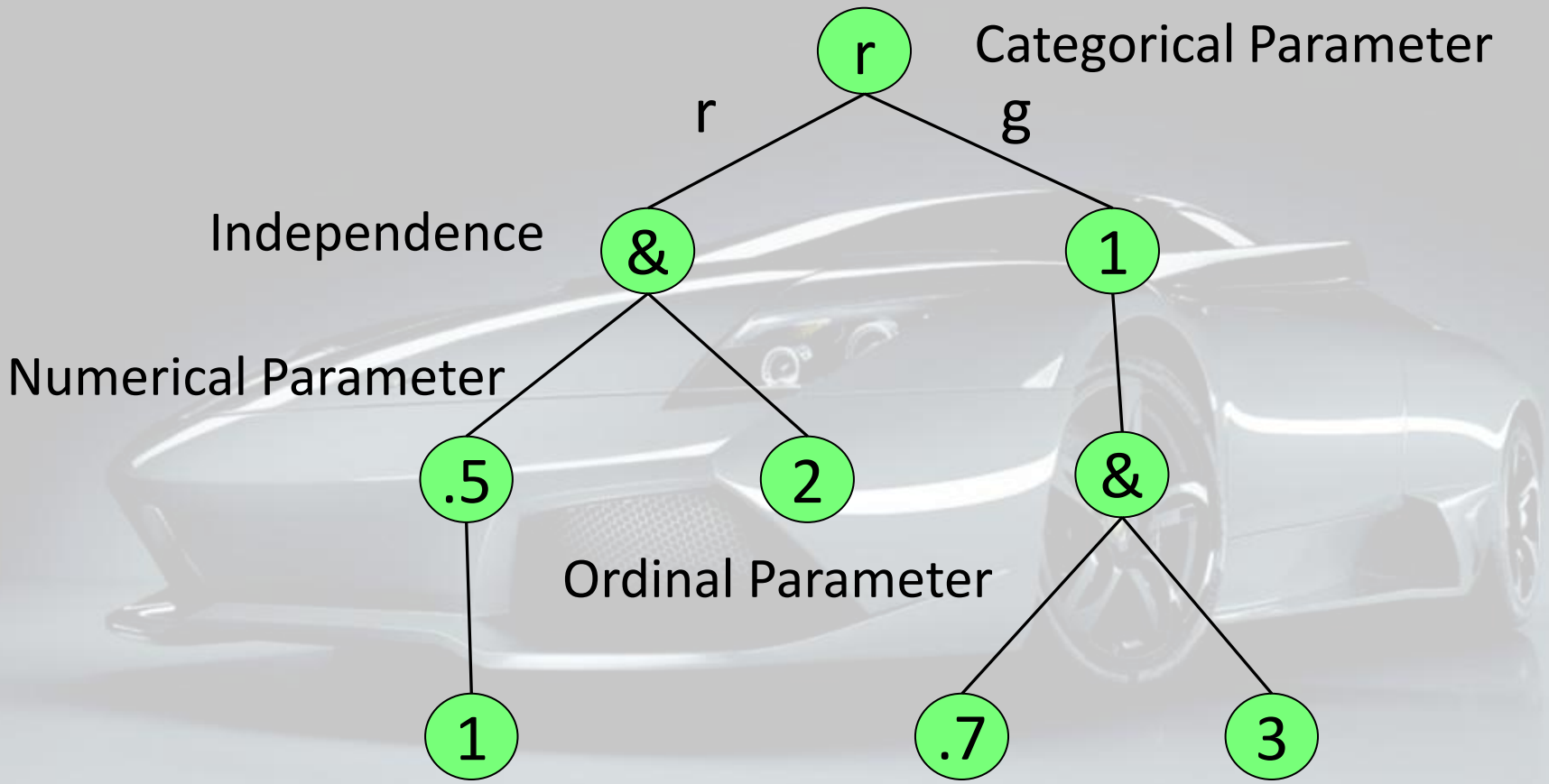
- General purpose tuner
- Handles various types of parameters
- Provides high-quality configurations
  - Robustly
  - With reasonable computational effort
- Exploits
  - Optimization technology
  - Parallelism

Content

- Instance-Oblivious Tuning
  - Overview of Approaches
  - **Parameters: Variable Tree Representation**
  - GGA: Gender-Based Genetic Algorithm
  - GGA: Numerical Results
- Algorithm Portfolios
  - Overview of Approaches
  - SATzilla
  - CP-Hydra
  - 3S

- Instance-Specific Tuning
  - Overview of Approaches
  - ISAC: Feature-based Parameter Selection
  - ISAC: Numerical Results

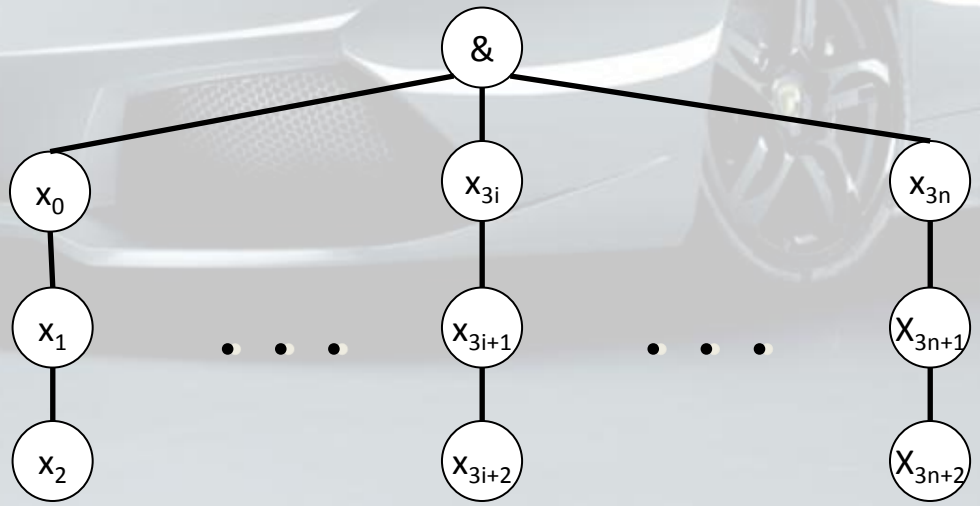
Variable Trees



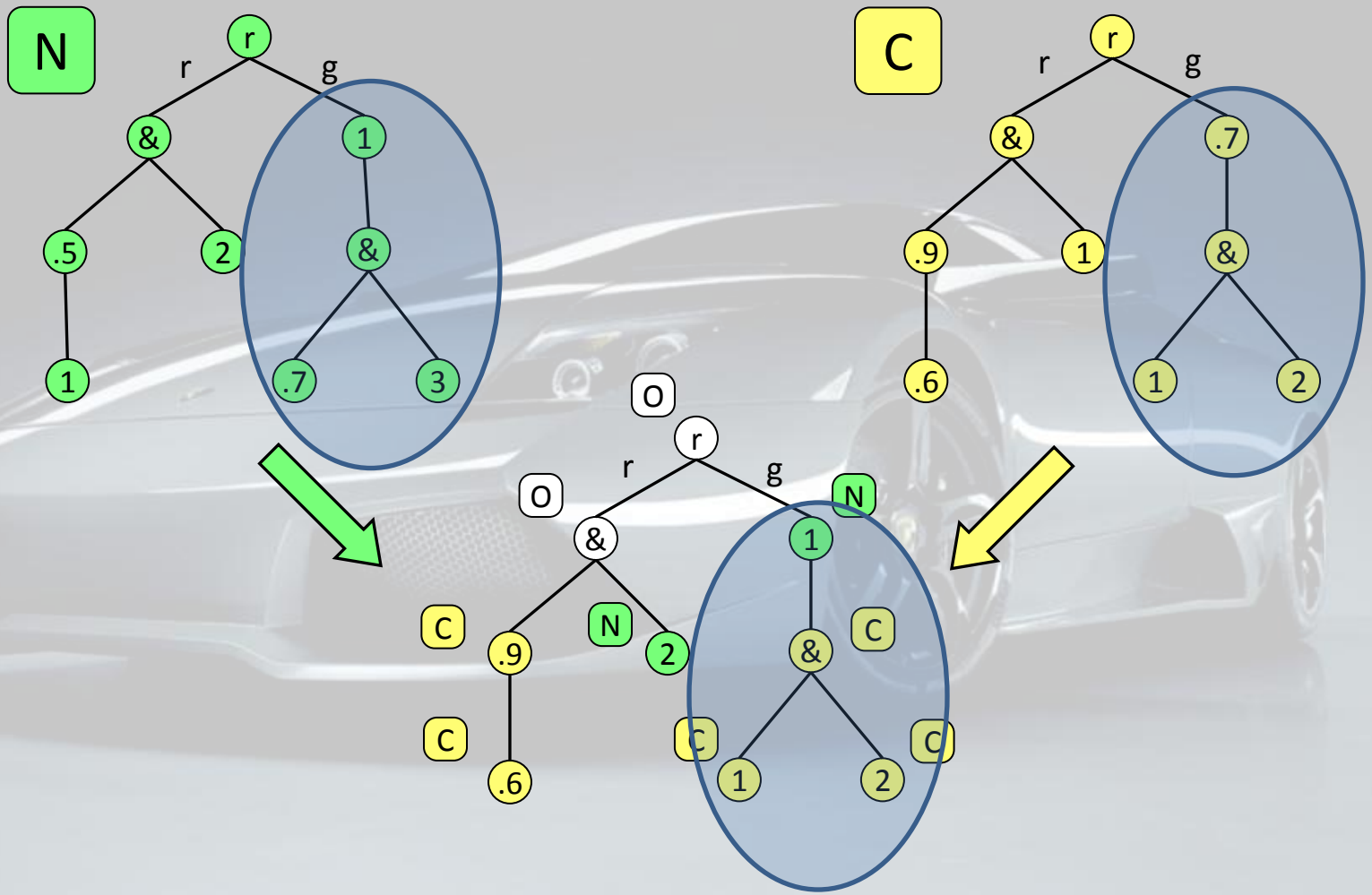
Variable Trees

- Parameter structure represented by an And-Or structure
- Represents parameter (in)dependence
- Example:

$$f(x) = \sum_{i \in [0,n]} (x_{3i} x_{3i+1} x_{3i+2} - q_i)^2$$

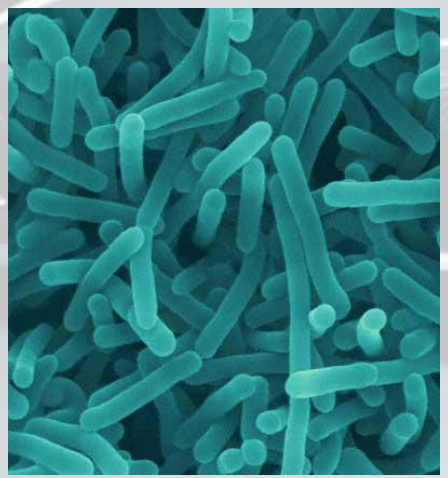


Generic Crossover Operator



Genetic Algorithm

- Computational Limitations
  - Low number of individuals
  - Low number of generations
- What did nature do when going from



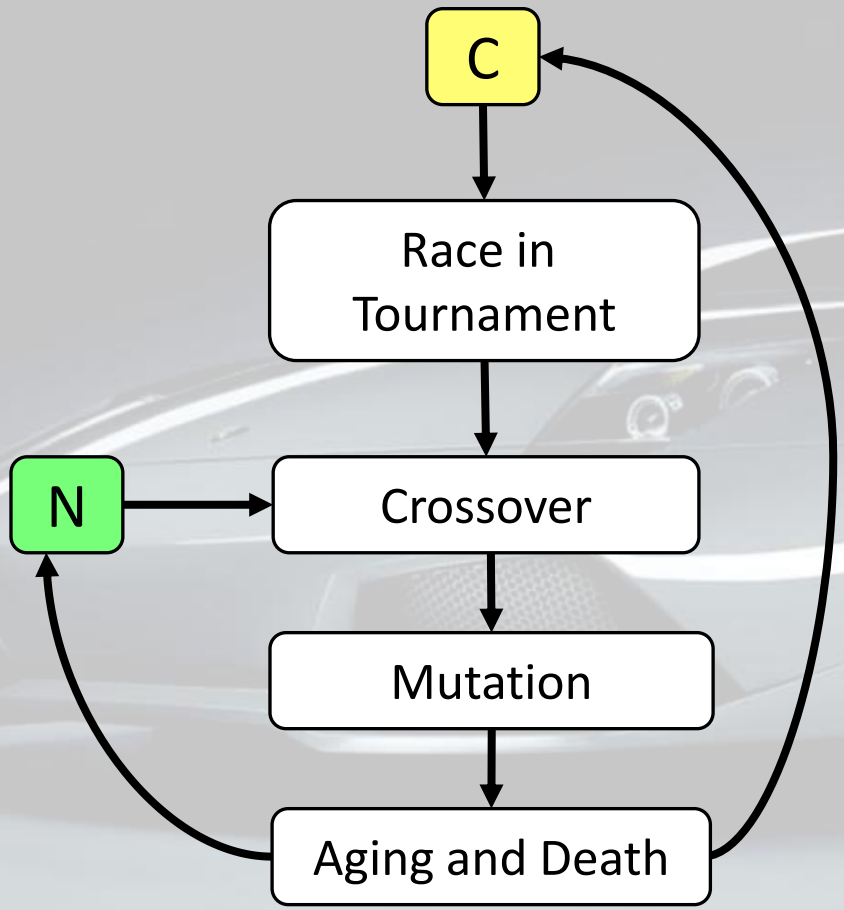
to

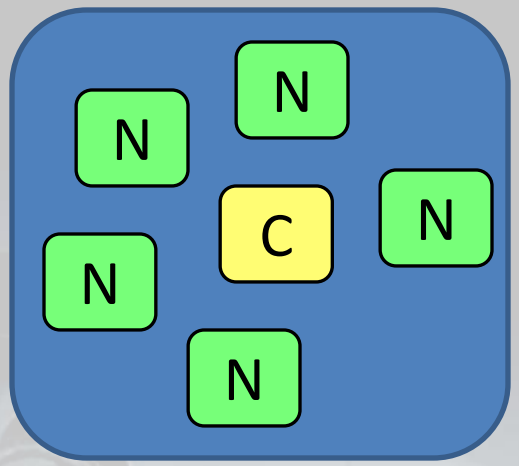
?

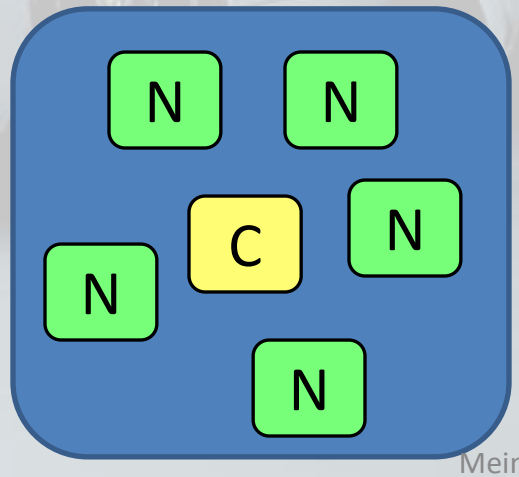
Gender-based Genetic Algorithm

- Optimization Problems
  - Low number of generations → aggressive optimization
  - Low number of individuals → emphasis on diversity
- How can genders help?
  - Split the population into two genders: competitive (C) and non-competitive (N)
  - Save 50% of evaluations
  - Racing: Winners determine evaluation time!
  - Can afford aggressive selection pressure on C
  - Individuals in N provide the needed diversity

Gender-based Genetic Algorithm

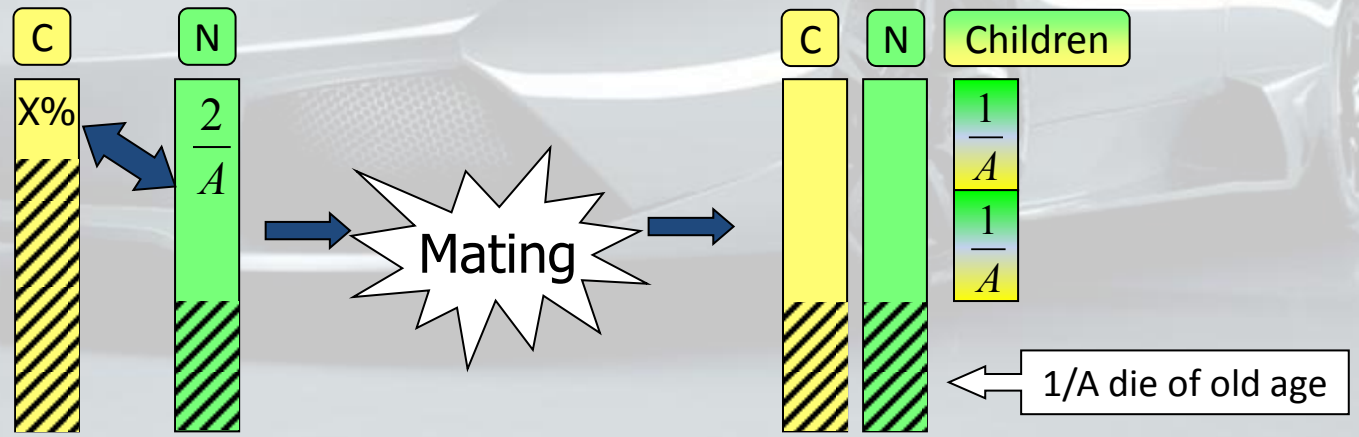






Population Control

- All members have an age
- Only 2/*A* of the N population mates
- 1/A of each population dies at age A



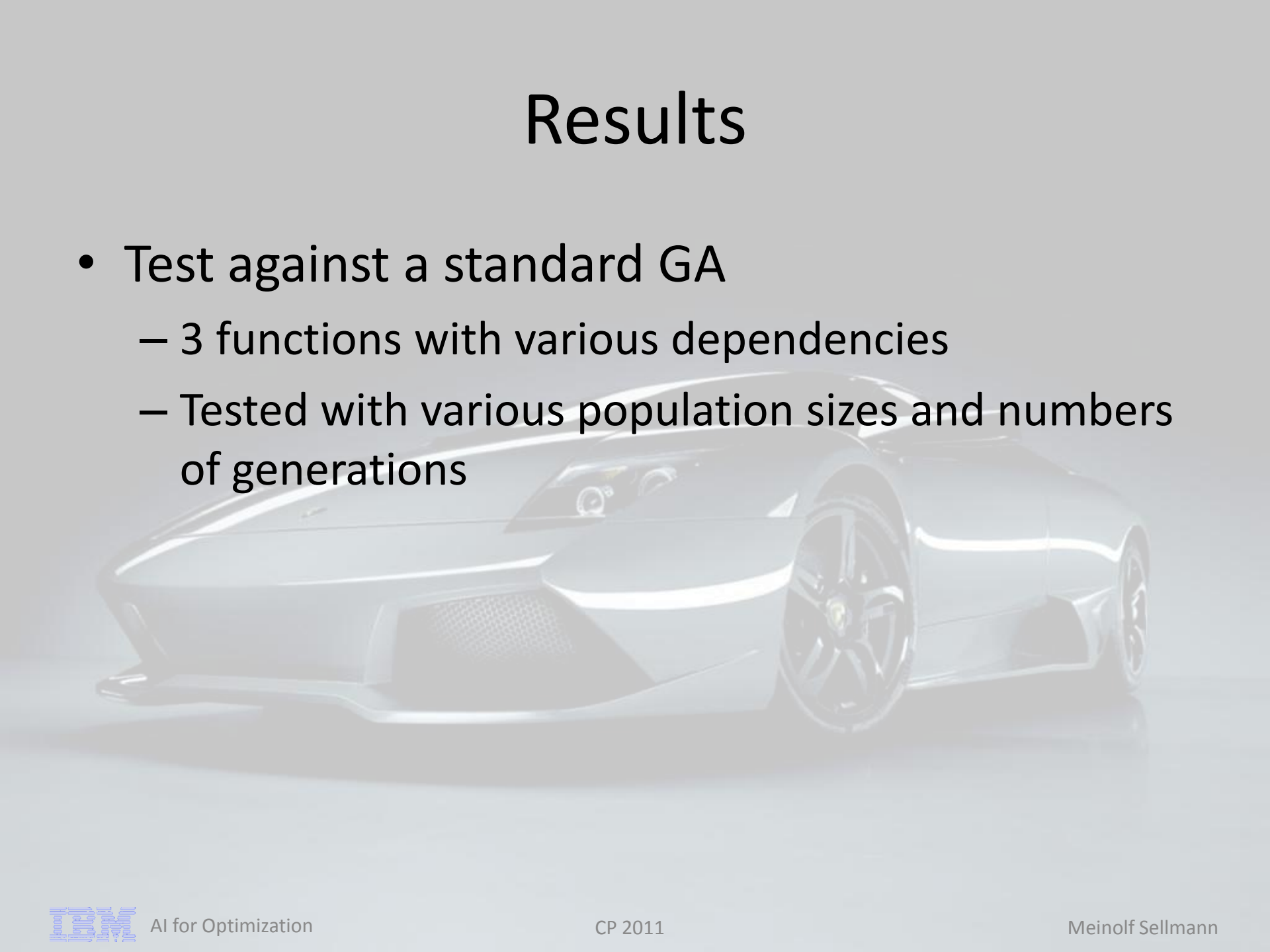
Content

- Instance-Oblivious Tuning
  - Overview of Approaches
  - Parameters: Variable Tree Representation
  - GGA: Gender-Based Genetic Algorithm
  - **GGA: Numerical Results**
- Algorithm Portfolios
  - Overview of Approaches
  - SATzilla
  - CP-Hydra
  - 3S

- Instance-Specific Tuning
  - Overview of Approaches
  - ISAC: Feature-based Parameter Selection
  - ISAC: Numerical Results

Results

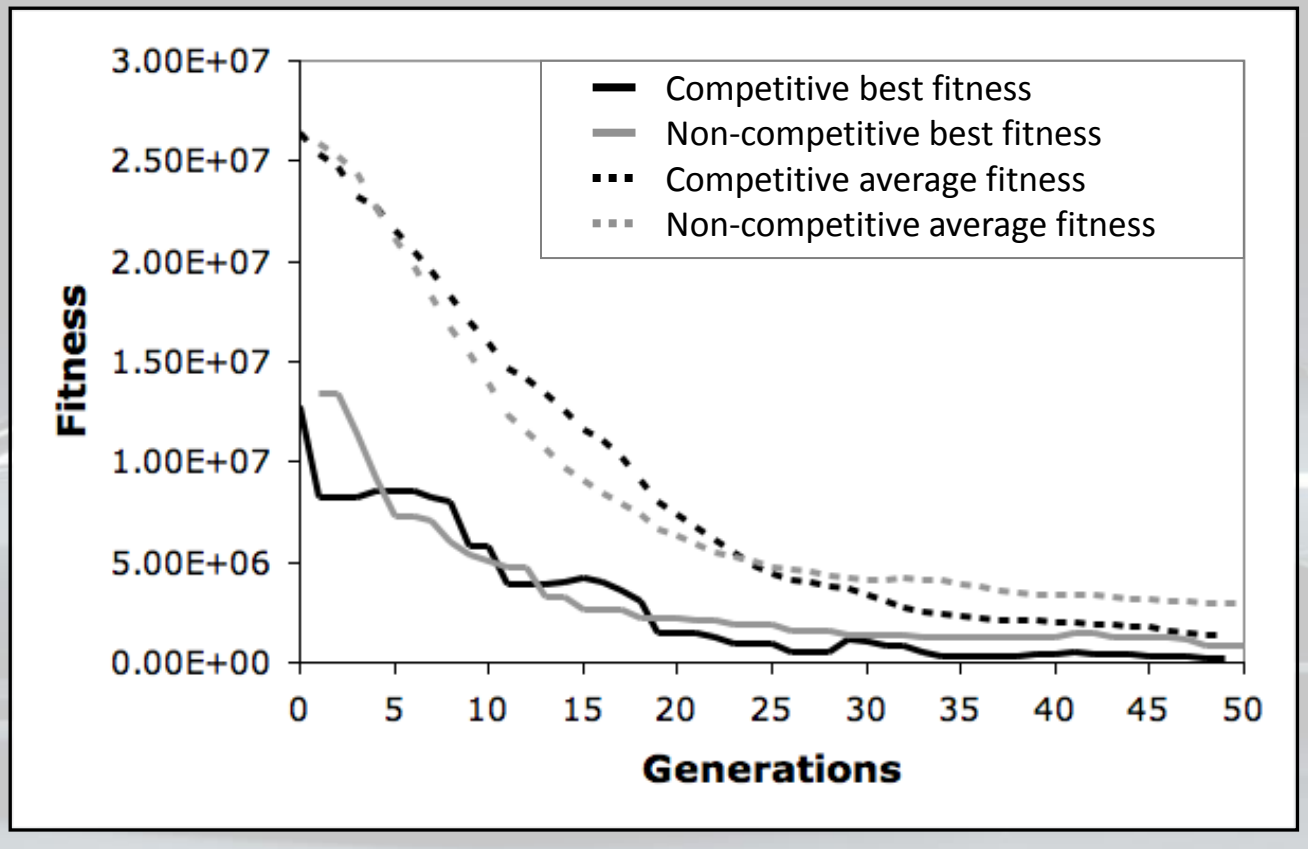
- Test against a standard GA
  - 3 functions with various dependencies
  - Tested with various population sizes and numbers of generations



Results

Prob-Pop-Gen	GGA IND	GGA DEP	GGA SEM	GA
f_1-1000-25	1.8(13.7)	29.8(13.7)	**1.44**(13.8)	7.19(26.0)
f_1-500-50	0.59(13.4)	28.9(12.8)	**0.4**(13.3)	4.56(25.5)
f_1-2000-25	1.54(27.1)	29.3(27.7)	**1.29**(27.6)	6.67(52.0)
f_1-1000-50	0.46(26.0)	28.7(26.1)	**0.31**(26.0)	4.41(51.0)
f_1-500-100	0.16(25.1)	28.3(25.0)	**0.13**(25.5)	2.69(50.5)
f_2-1000-25	1442(13.7)	6075(13.8)	**1229**(13.4)	4962(26.0)
f_2-500-50	392(13.3)	5273(13.0)	**340**(13.0)	3127(25.5)
f_2-2000-25	**1104**(27.4)	5121(25.3)	1119(27.6)	4405(52.0)
f_2-1000-50	307(25.6)	5886(27.5)	**304**(25.8)	3184(51.0)
f_2-500-100	141(25.3)	4830(25.4)	**124**(26.0)	2002(50.5)
f_3-1000-25	**16.2**(13.6)	18.5(13.9)	16.4(13.7)	43.2(26.0)
f_3-500-50	**5.75**(13.2)	8.01(13.5)	6.33(13.7)	31.9(25.5)
f_3-2000-25	14.7(27.5)	15.9(27.5)	**13.7**(27.2)	41.3(52.0)
f_3-1000-50	**4.86**(25.9)	6.40(26.5)	5.27(25.8)	31.8(51.0)
f_3-500-100	1.33(26.5)	1.7(27.1)	**1.25**(25.8)	22.5(50.5)

Results


3.00E+07
2.50E+07
2.00E+07
1.50E+07
1.00E+07
5.00E+06
0.00E+00
Fitness
0
5
10
15
20
25
30
35
40
45
50
Generations
Competitive best fitness
Non-competitive best fitness
Competitive average fitness
Non-competitive average fitness

Results

	GA		GGA		Improvement
	Avg.	Std. Dev	Avg.	Std. Dev.	[%]
Time	23.5K	4.48K	926	1.12K	96
Quality	1.79	1.57	0.07	0.01	96

- Standard GA vs. GGA tuning SAT-solver SAPS
- 40 generations with 30 members
- Cutoff of 10 seconds
- GA wastes lots of time on bad solutions

Results

[ms]	ParamILS		GGA	
Run	Train	Test	Train	Test
1	51.55	52.12	37.25	35.44
2	53.91	51.45	46.15	41.69
3	55.31	50	55.85	49.52
4	55.11	51.8	36.07	33.4
5	53.72	50.91	46.15	40.1
6	54.96	52.4	35.68	33.56
7	54.67	52.79	34.69	32.2
8	55.11	49.91	37.54	32.76
9	56.24	51.09	38.24	35.42
10	56.26	51.31	34.7	33.53
11	55.02	52.26	35.99	34.52
12	54.29	51.61	36.1	33.99
13	54.31	51.42	36.35	33.59
14	56.58	52.31	37.42	34.58
15	57.38	54.15	38.79	36.66
16	57	54.09	52.98	52.25
17	56.31	52.6	35.78	32.68
18	58	53.73	38.59	35.06
19	54.52	51.83	35.83	33.9
20	61.47	55.85	36.57	34.56
⌀	55.6	52.2	39.3	36.5
σ	2.0	1.44	6.0	5.5

SAPS (ms)

SAT4J (s)

[s]	ParamILS		GGA	
Run	Train	Test	Train	Test
1	3.65	0.99	1.07	1.07
2	1.06	1.07	1.05	1.06
3	1.07	1.07	1.06	1.06
4	0.99	1.05	1.06	1.07
5	5.11	2.22	1.07	1.14
6	1.04	1.04	3.20	3.90
7	5.29	2.31	1.05	1.05
8	6.27	2.38	1.06	1.06
9	1.05	0.53	1.04	1.05
10	1.06	1.07	1.05	1.05
11	1.17	1.06	1.14	1.06
12	1.04	1.05	1.05	1.13
13	1.05	1.06	1.07	1.07
14	1.05	1.05	1.05	1.06
15	1.04	1.05	1.05	1.06
16	6.34	6.83	1.08	1.08
17	5.17	5.35	1.07	1.07
18	4.70	4.32	1.05	1.06
19	5.57	5.20	1.06	1.06
20	5.03	4.97	1.05	1.06
⌀	2.94	2.28	1.17	1.21
σ	2.2	1.92	0.48	0.63

Results

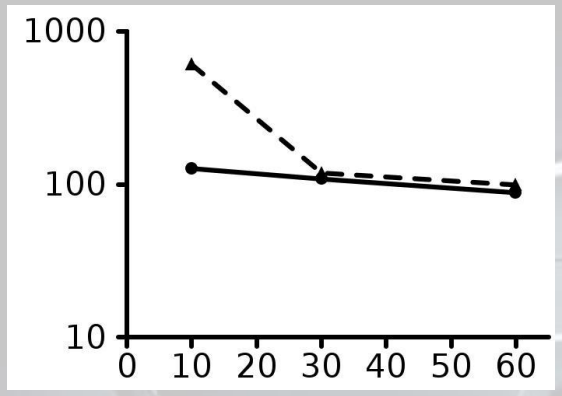
- Target algorithms: SAPS, SPEAR, SAT4J, SAT4J*

Solver	ParamILS	GGA	%Imprv.	Welsh's T-Value
SAPS (ms)	52.2 (1.44)	36.5 (5.5)	31.30	<0.01
SPEAR (s)	1.49 (0.087)	1.50 (0.077)	-0.67	0.33
SAT4J (s)	2.38 (1.97)	1.29 (0.76)	45.80	0.01
SAT4J* (s)	3.74 (1.28)	3.20 (0.81)	14.4	0.04

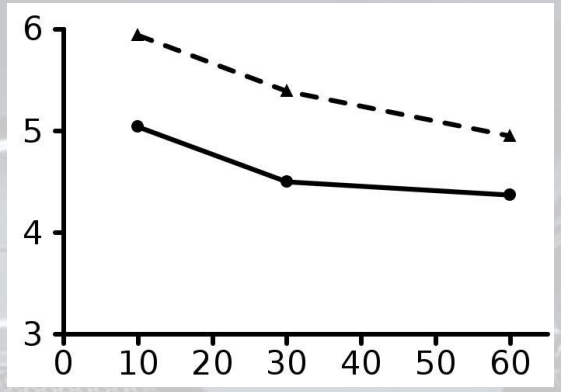
Test performance (average, variance) after 20 executions

Results

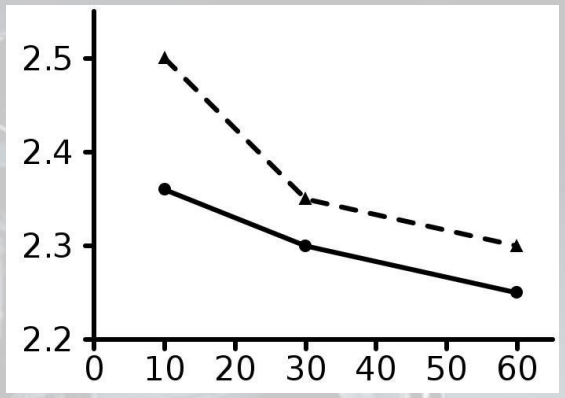
SAPS



SAT4J



SPEAR



Tuning Quality over Time

Content

- Instance-Oblivious Tuning
  - Overview of Approaches
  - Parameters: Variable Tree Representation
  - GGA: Gender-Based Genetic Algorithm
  - GGA: Numerical Results
- **Algorithm Portfolios**
  - Overview of Approaches
  - SATzilla
  - CP-Hydra
  - 3S

- Instance-Specific Tuning
  - Overview of Approaches
  - ISAC: Feature-based Parameter Selection
  - ISAC: Numerical Results

Algorithm Portfolios

Overview of Existing Methods

[from Smith-Miles 2009]

Table II. Recasting the Literature Using Rice's Framework

Reference	Domain	P	A	Y	F	S
Aha [1992]	classification	75 datasets	3 rule learners	accuracy	7 measures of Statistical distributions	C4.5
Brazdil & Henery [1994]	classification	22 datasets = StatLog	23 machine learning, neural and statistical learning algorithms = StatLog	accuracy	16 measures of statistical distributions and information theory = StatLog	C4.5
Gama & Brazdil [1995]	classification	StatLog	StatLog	accuracy	StatLog	regression
Linder & Studer [1999]	classification	StatLog	StatLog	accuracy	StatLog	CBR
Kalousis & Theoharis [1999]	classification	StatLog	StatLog	accuracy	StatLog	NOEMON (pairwise NNs)
Brazdil et al. [2003]	classification	53 datasets	10 machine learning algorithms	accuracy and time	StatLog	K-nearest neighbor ranking
Lim et al. [2000]	classification	32 datasets	33 machine learning algorithms	accuracy, time, tree size	StatLog	C4.5
Smith et al. [2001]	classification	57 datasets	6 machine learning algorithms	accuracy	21 measures: StatLog plus additional statistical measures	Neural network
Smith et al. [2002]	classification	57 datasets	6 machine learning algorithms	accuracy	21 measures: StatLog plus additional statistical measures	Self-organizing feature map
Ali & Smith [2006]	classification	112 datasets	8 machine learning algorithms	accuracy and time	31 measures: StatLog plus additional statistical measures	C4.5
Kopf et al. [2000]	regression	5450 datasets	3 regression models	error	StatLog	C5.0
Ali & Smith-Miles [2006]	kernel selection	112 datasets	5 kernels within SVM	accuracy	29 measures: StatLog plus additional statistical measures	C4.5
Soares et al. [2004]	SVM parameter selection	42 datasets	11 parameters	error	14 statistical features of regression problems	K-nearest neighbor ranking
Arinze et al. [1997]	forecasting	67 time series	6 forecasting methods	error	6 features	Expert system

Overview of Existing Methods

[from Smith-Miles 2009]

Venkat. & Sohl [1999]	forecasting	180 series	9 forecasting methods	error	6 features	Neural network
Prudencio & Ludermir [2004]	forecasting	99 series	2 forecasting methods	error	14 features	C4.5
Prudencio & Ludermir [2004]	forecasting	645 series	3 forecasting methods	error	5 features	NOEMON
Wang [2005]	forecasting	315 series	4 forecasting methods	error	13 features	C4.5 and SOFM
Lagoudakis et al. [2001]	sorting	10000 sequences	3 sorting algorithms	time	1 feature (length of sequence)	Dynamic programming
Guo [2003]	sorting	43195 sequences	5 sorting algorithms	time	3 measures of presortedness	C4.5 & Bayesian classifier
Horvitz et al. [2001]	constraint satisfaction	5000 QWH problems	2 solvers	time	Statistical features of the problem and solver statistics	Bayesian network
Leyton-Brown et al. [2002]	constraint satisfaction	8544 WDP problems	1 solver (CPLEX)	time	25 Statistical features of the problem and solver statistics	Regression
Leyton-Brown et al. [2003]	constraint satisfaction	4500 WDP problems	3 solvers	time	25 Statistical features of the problem and solver statistics	Regression
Nudelman et al. [2004]	constraint satisfaction	20000 SAT problems	3 solvers	time	91 Statistical features of the problem and solver statistics	Regression
Xu [2007]	constraint satisfaction	4811 SAT problems	7 solvers with & w/o preprocessing	time	48 features of the problem and solver statistics	Regression
Samulowitz [2007]	constraint satisfaction	1697 QBF problems	10 branching heuristics	time	78 features of the problem and solver statistics	Regression
Smith-Miles [2008]	optimization	28 QAP problems	3 meta-heuristics	objective function gap	8 measures of problem size and complexity, and search space characteristics	Neural network and SOFM

Overview of Existing Methods

- Adaptive Methods
  - Reactive Tabu Search [Battiti and Tecchiolli, ’94]
  - STAGE [Boyan and Moore, ’00]
  - Impact Based Search Strategies [Philippe Refalo, ’04]
  - Disco-Novo-GoGo: [Ansotegui et al, ’06]
- Algorithm Portfolios
  - Parallel Execution [Gomes and Selman, ’01]
  - SATzilla [Xu et al., ’07]
  - QBF Tuner [Samulowitz et al, ‘07]
  - CP-Hydra [O’Mahony et al, ‘08]
  - Latent Class Model Portfolio [Silverthorn et al, ‘10]
  - SAT Solver Selector [Samulowitz et al, ‘11]

SATzilla

- Empirical hardness model for each solver
  - Trained offline
  - Based on linear regression
- At runtime, choose solver with shortest predicted runtime!
- Most successful portfolio approach to date [SAT Competition Gold Medallist ‘07 and ‘09]

CP Hydra

- Solver Scheduler (instead of Solver Selector)
- Choose 10 nearest neighbors of given instance
- Use MIP to compute schedule that solves most neighbors within time-limit.

$$\begin{aligned}
\min \quad & (C+1)\sum_i y_i + \sum_{S,t} t x_{S,t} && && (1)\\
s.t. \quad & y_i + \sum_{(S,t) \mid i \in V_{S,t}} x_{S,t} \geq 1 && \forall i && (2)\\
& \sum_{S,t} t x_{S,t} \leq C && && (3)\\
& y_i, x_{S,t} \in \{0,1\} && \forall i, S, t && (4)
\end{aligned}$$

Content

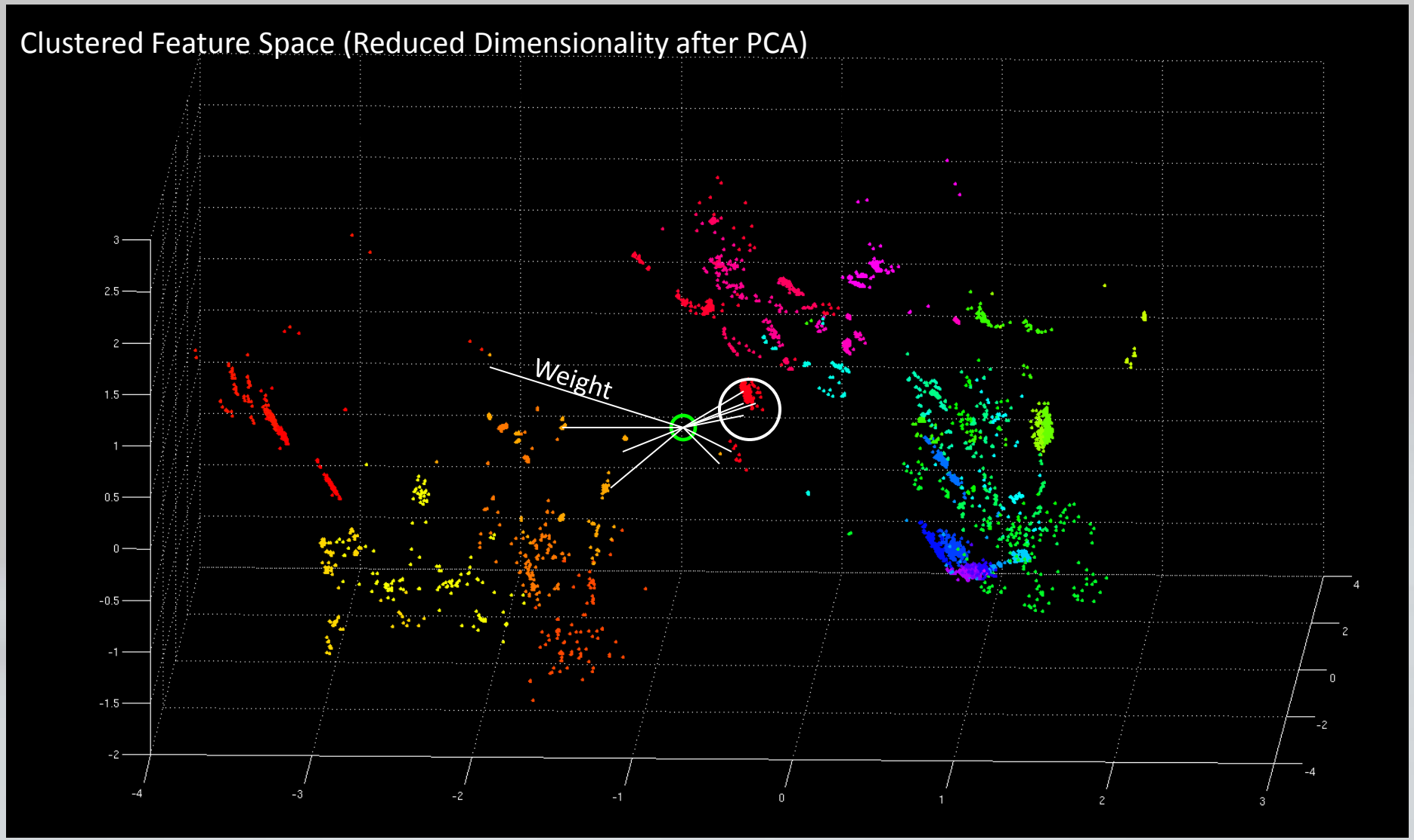
- Instance-Oblivious Tuning
  - Overview of Approaches
  - Parameters: Variable Tree Representation
  - GGA: Gender-Based Genetic Algorithm
  - GGA: Numerical Results
- Algorithm Portfolios
  - Overview of Approaches
  - SATzilla
  - CP-Hydra
  - **3S**

- Instance-Specific Tuning
  - Overview of Approaches
  - ISAC: Feature-based Parameter Selection
  - ISAC: Numerical Results

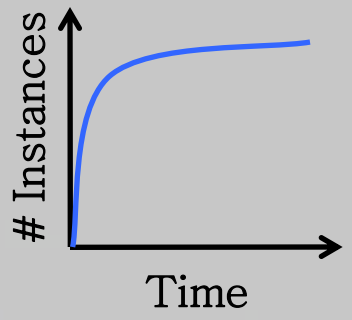
SAT Solver Selector

- Highly diverse set of base solvers (local search, tree-search, learning solvers, etc)
- Based on AI: Non-model-based machine learning to select a “good” solver
- Based on OR: Math programming to schedule solvers

SAT Solver Selector



Scheduling



- Runtime distribution of SAT solvers
  - High percentage of instances solved quickly
  - Instances solved quickly vary a lot by solver
- Computing Instance-oblivious Schedule
  - Given a time budget
  - Decide which solvers to run and for how long
  - Can be formulated as MIP

$$
\begin{aligned}
\min \quad & (C+1)\sum_i y_i + \sum_{S,t} t x_{S,t} & \\
s.t. \quad & y_i + \sum_{(S,t) \mid i \in V_{S,t}} x_{S,t} \geq 1 & \forall i \\
& \sum_{S,t} t x_{S,t} \leq C & \\
& y_i, x_{S,t} \in \{0,1\} & \forall i, S, t
\end{aligned}
$$

Empirical Evaluation

CPU time						
	SAT+UNSAT		SAT		UNSAT	
Rank	Solver	#	Solver	#	Solver	#
Ref	SATzilla_R	384	TNM	324	march_hi	123
1	3S (coach)	408	sparrow2011	362	march_rw	123
2	ppfolio //	377	sattime2011	334	mphasesat_m	104
3	ppfolio seq	375	eagleup	328	ppfolio //	101

SAT Competition 2011: Random Category

Empirical Evaluation

CPU time						
	SAT+UNSAT		SAT		UNSAT	
Rank	Solver	#	Solver	#	Solver	#
Ref	clasp	149	clasp	85	clasp	64
1	3S (coach)	163	ppfolio //	120	clasp	62
2	ppfolio //	155	ppfolio seq	114	3S (coach)	51
3	ppfolio seq	152	3S (coach)	112	glucose	42

SAT Competition 2011: Crafted Category

Content

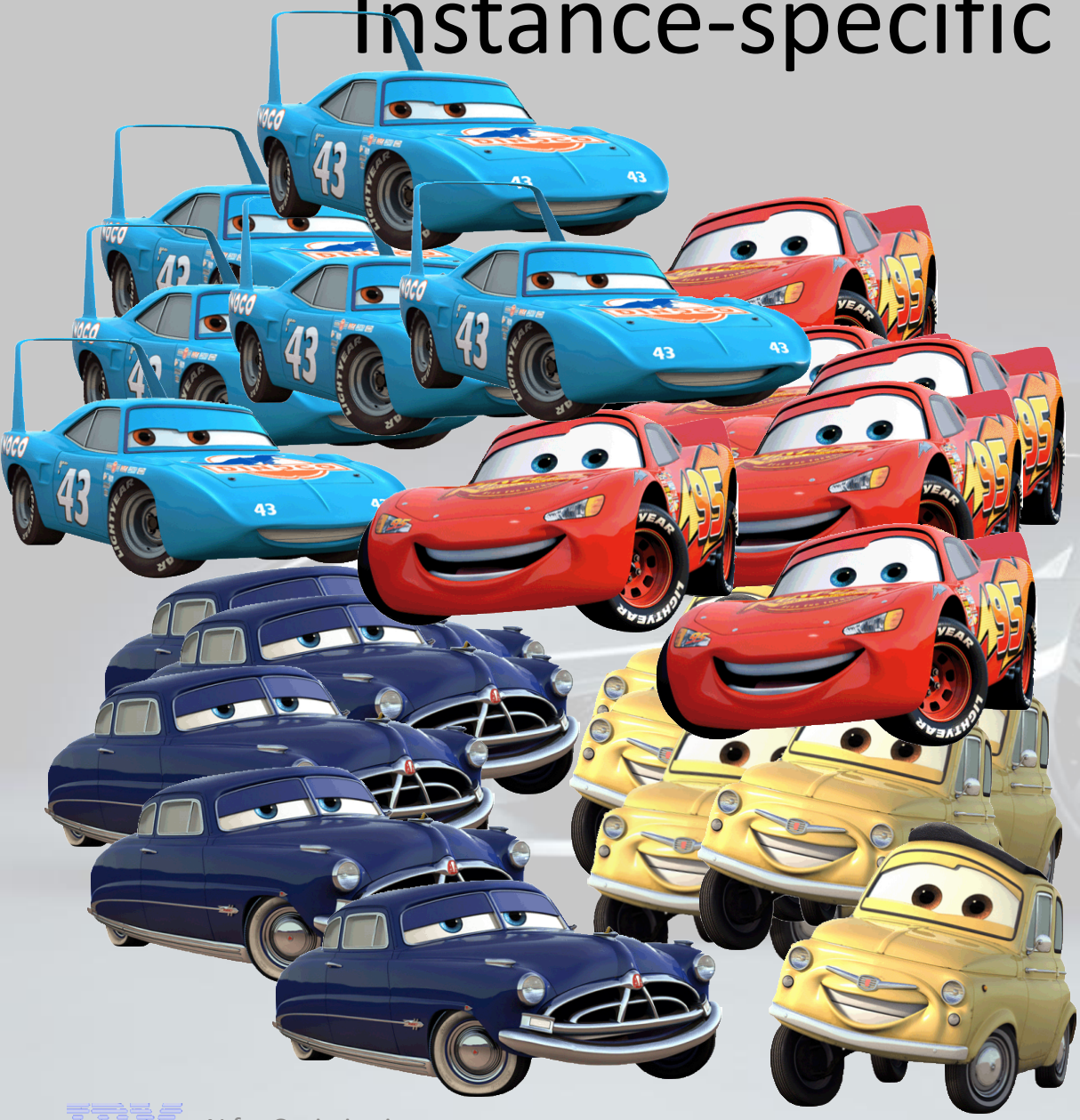
- Instance-Oblivious Tuning
  - Overview of Approaches
  - Parameters: Variable Tree Representation
  - GGA: Gender-Based Genetic Algorithm
  - GGA: Numerical Results
- Algorithm Portfolios
  - Overview of Approaches
  - SATzilla
  - CP-Hydra
  - 3S

- **Instance-Specific Tuning**
  - Overview of Approaches
  - ISAC: Feature-based Parameter Selection
  - ISAC: Numerical Results

Instance-oblivious Tuning

Algorithm Portfolios

Instance-specific Tuning

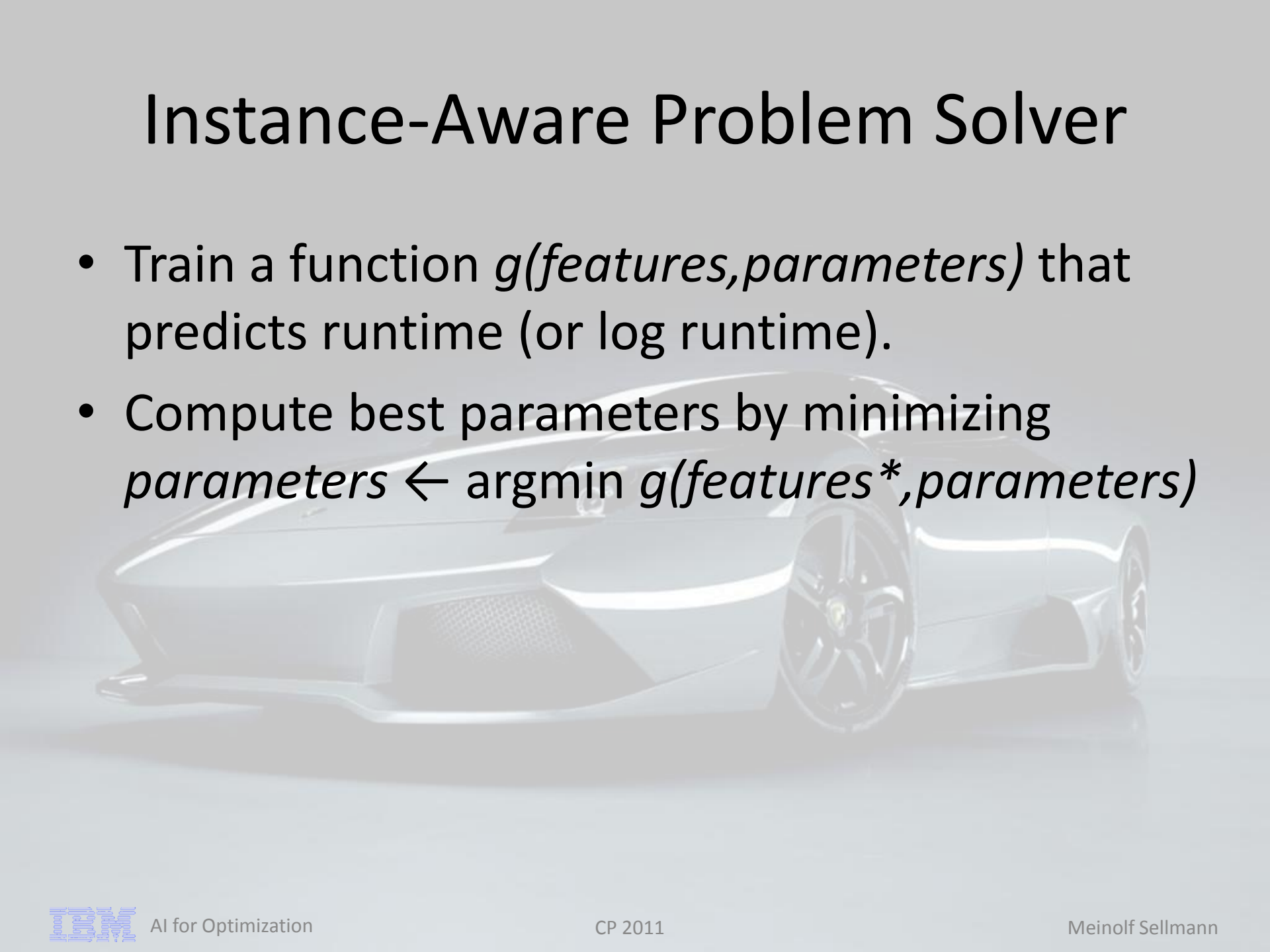


Overview of Existing Methods

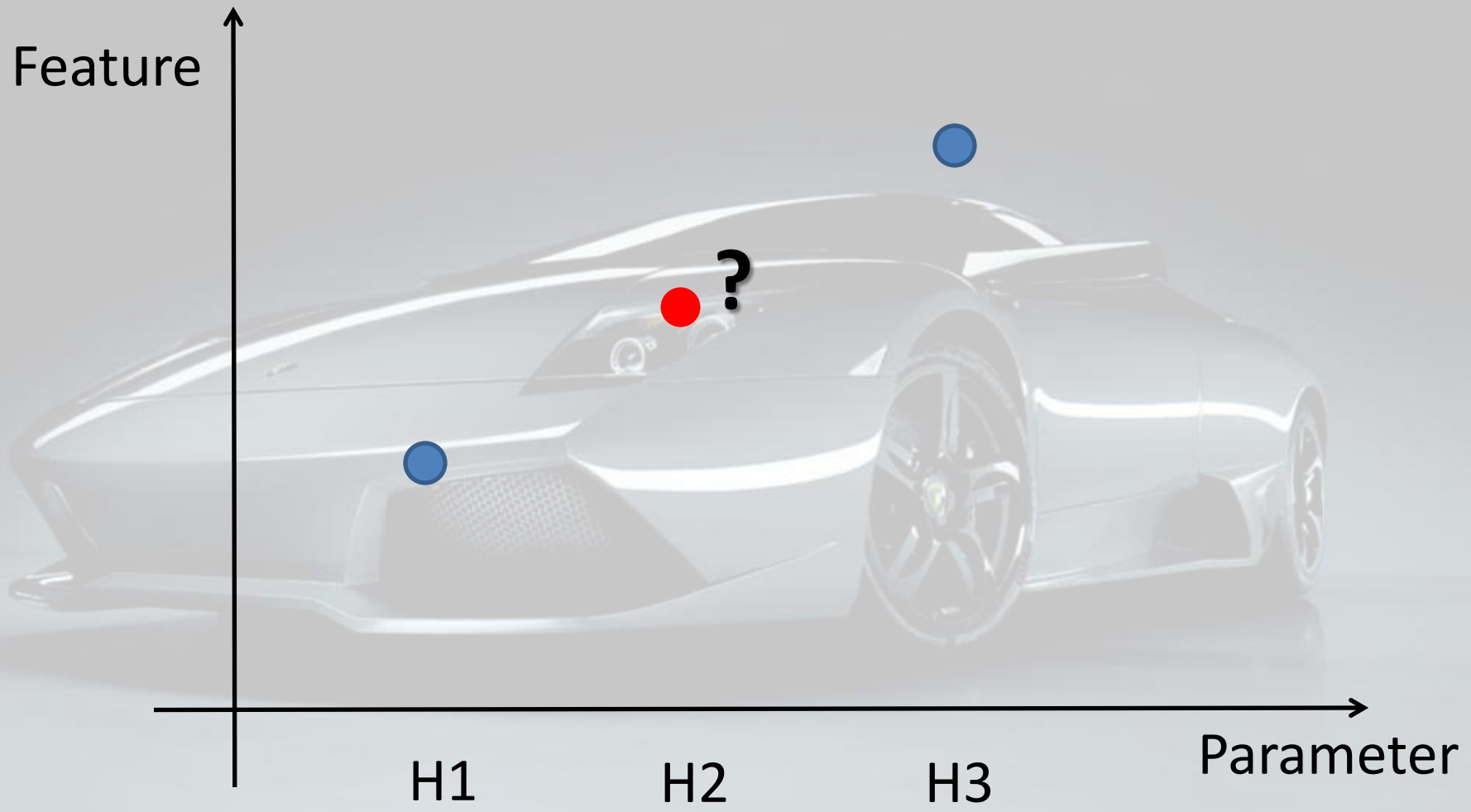
- Model-based Tuners
  - Instance-Aware Problem Solver [Hutter et al, ’05]
  - Hydra [**NOT** CP-Hydra! Xu et al, ‘10]
- Non-Model-Based
  - ISAC [Kadioglu et al, ‘10]

Instance-Aware Problem Solver

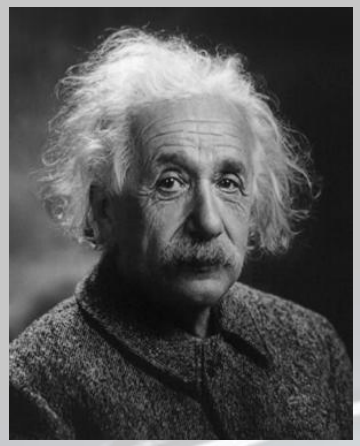
- Train a function *g(features,parameters)* that predicts runtime (or log runtime).
- Compute best parameters by minimizing *parameters* ← argmin *g(features*,parameters)*

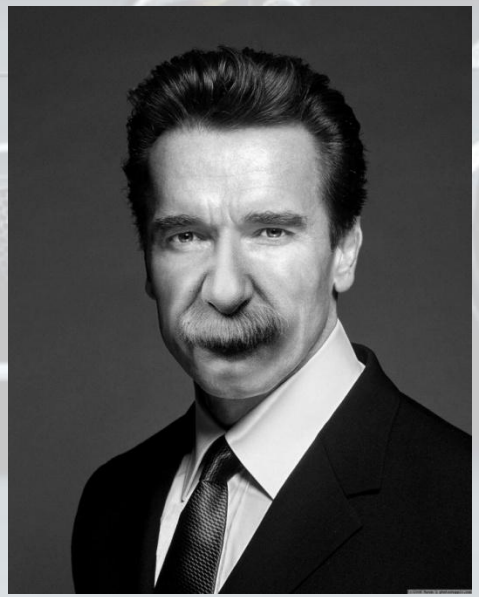


Instance-Aware Problem Solver

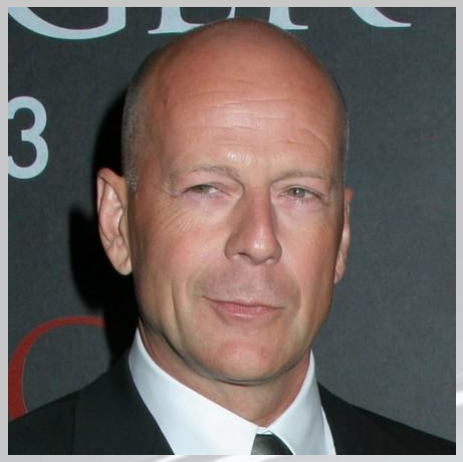


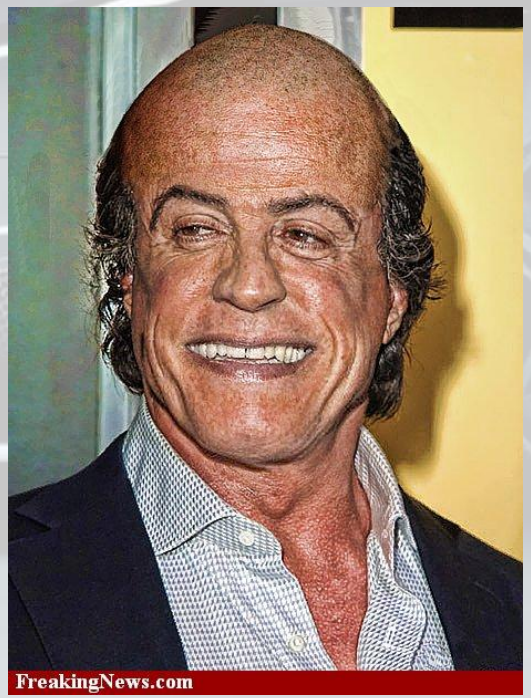
Instance-Aware Problem Solver





Instance-Aware Problem Solver





Hydra

- Combines SATzilla and ParamILS
- Repeat
  - For each instance, set timeout to min-time in current portfolio
  - Find a new parameter setting with ParamILS
  - Add it to the portfolio

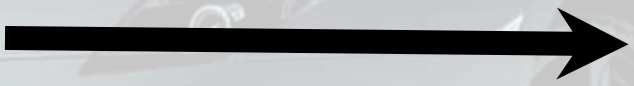
ISAC: Instance-Specific Algorithm Configuration

- Objectives
  - Adjust parameters to inputs
  - Learn a mapping from input features to parameter settings
  - Account for high computational costs
  - Exploit parallelism
- Approach
  - Cluster inputs!
  - Compute configuration for each cluster with GGA
  - At runtime
    - Determine nearest cluster
    - Use corresponding configuration
  - Inherently parallel, no interpolation, no untested configurations, bias limited to feature metric

Normalization

3	900.25
3.875	900.25
3.875	776.5
10	776.5
9.125	1024
9.125	900.25
8.25	776.5
7.375	157.75
8.25	34

	X1	X2
min	3	34
max	10	1024



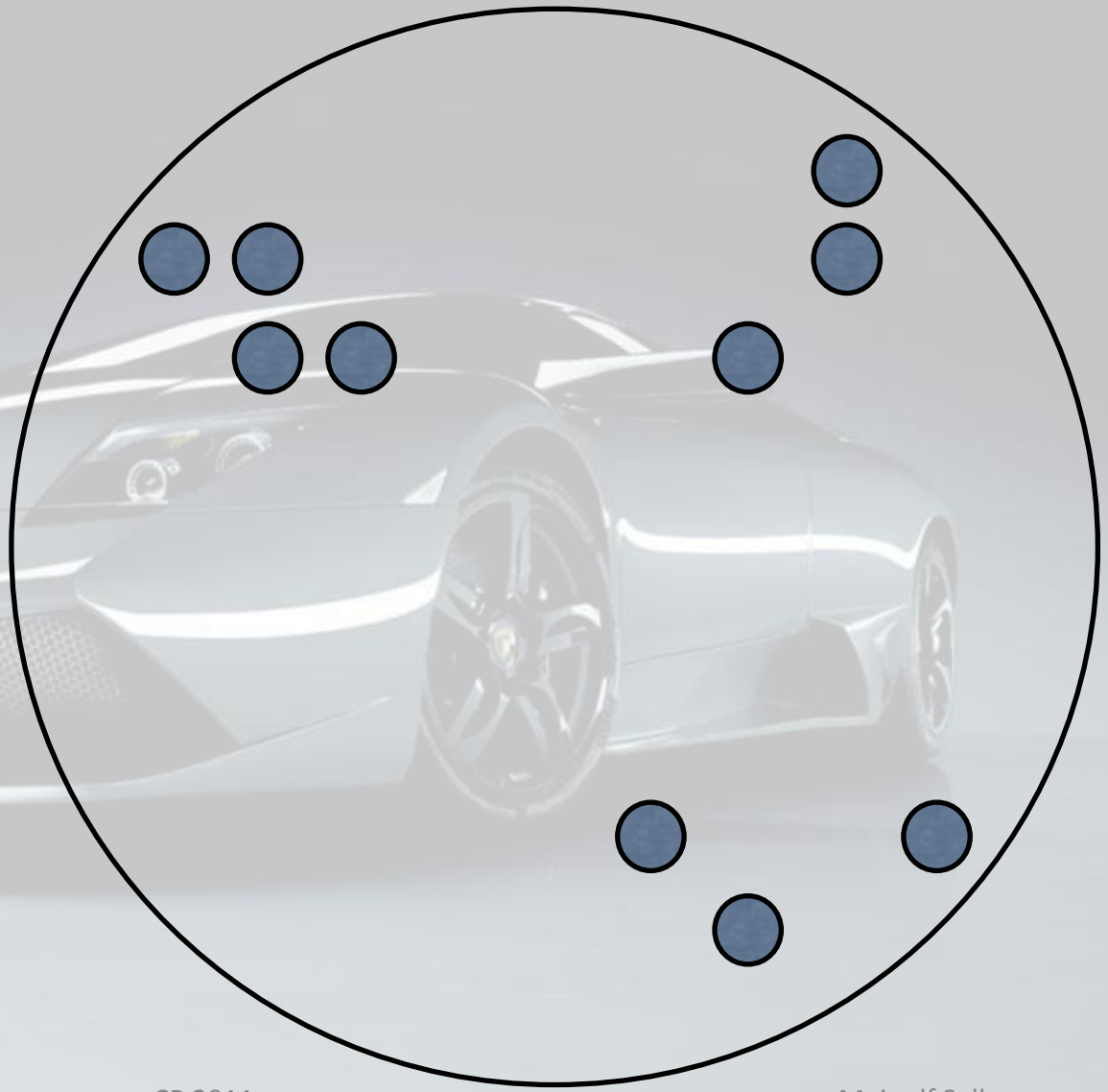
$X1 \rightarrow (2X1-13)/7$

$X2 \rightarrow (2X2-1058)/990$

-1.00	0.75
-0.75	0.75
-0.75	0.50
1.00	0.50
0.75	1.00
0.75	0.75
0.50	0.50
0.25	-0.75
0.50	-1.00

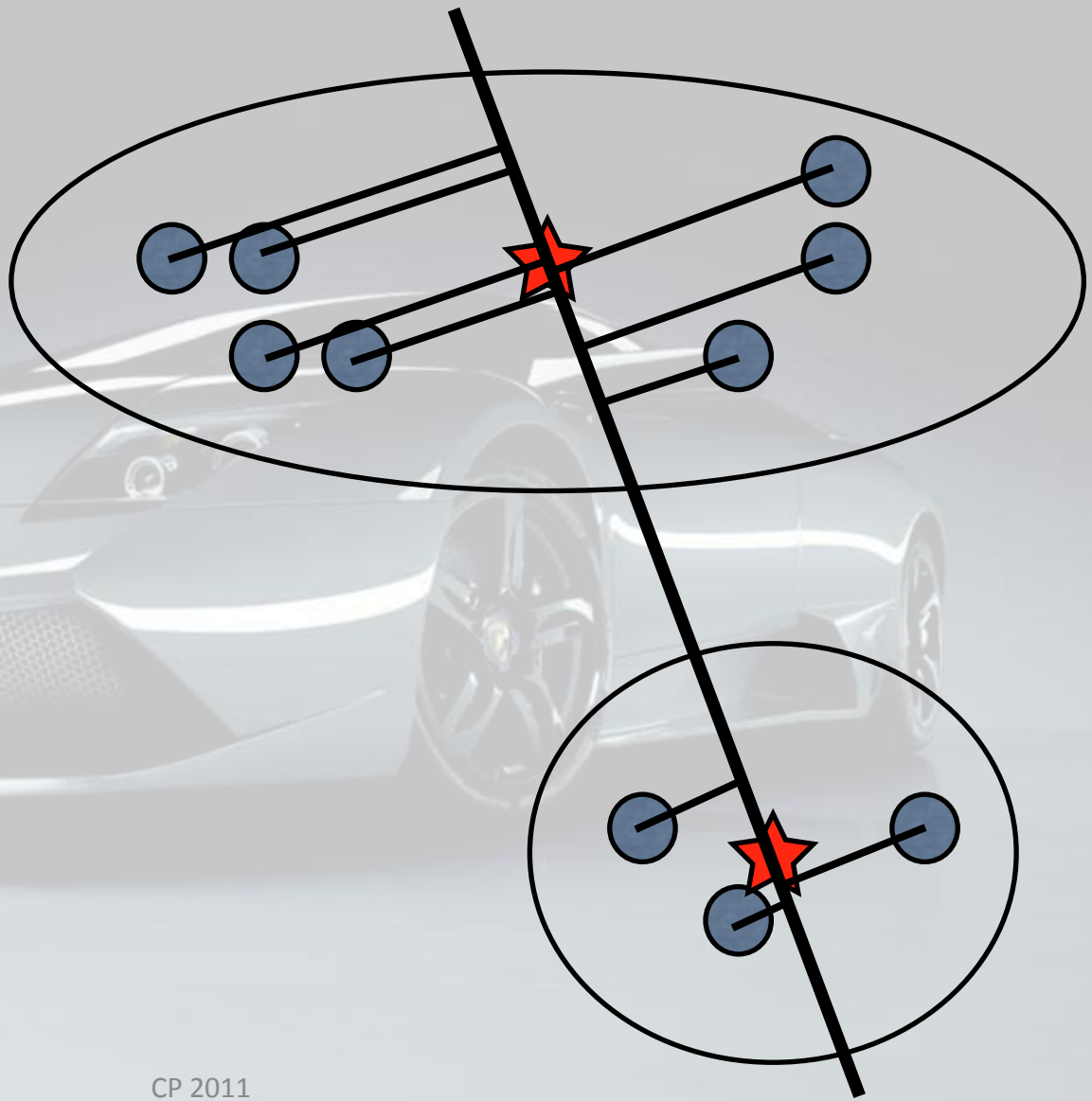
Approach - Clustering

-1.00	0.75
-0.75	0.75
-0.75	0.50
-0.50	0.50
0.75	1.00
0.75	0.75
0.50	0.50
0.25	-0.75
0.50	-1.00



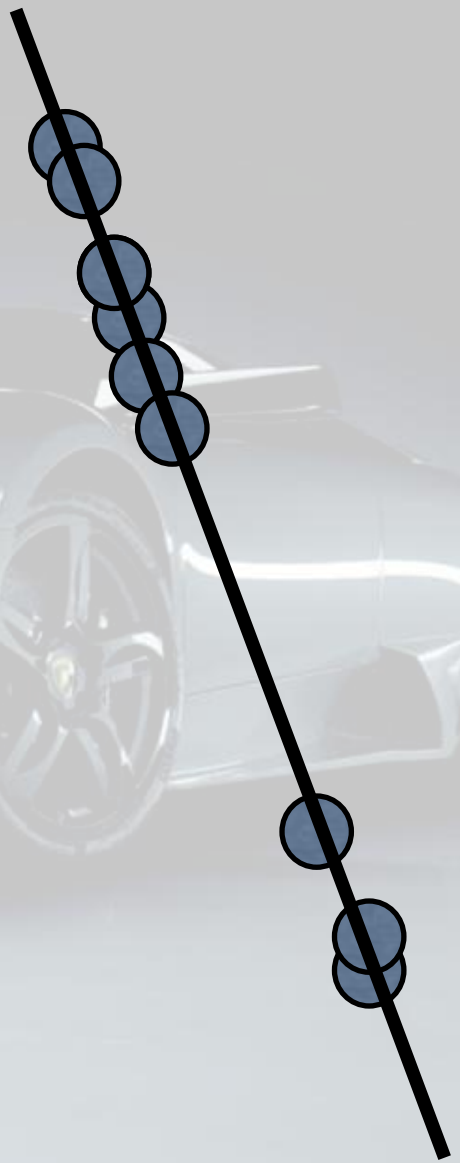
Approach - Clustering

-1.00	0.75
-0.75	0.75
-0.75	0.50
-0.50	0.50
0.75	1.00
0.75	0.75
0.50	0.50
0.25	-0.75
0.50	-1.00



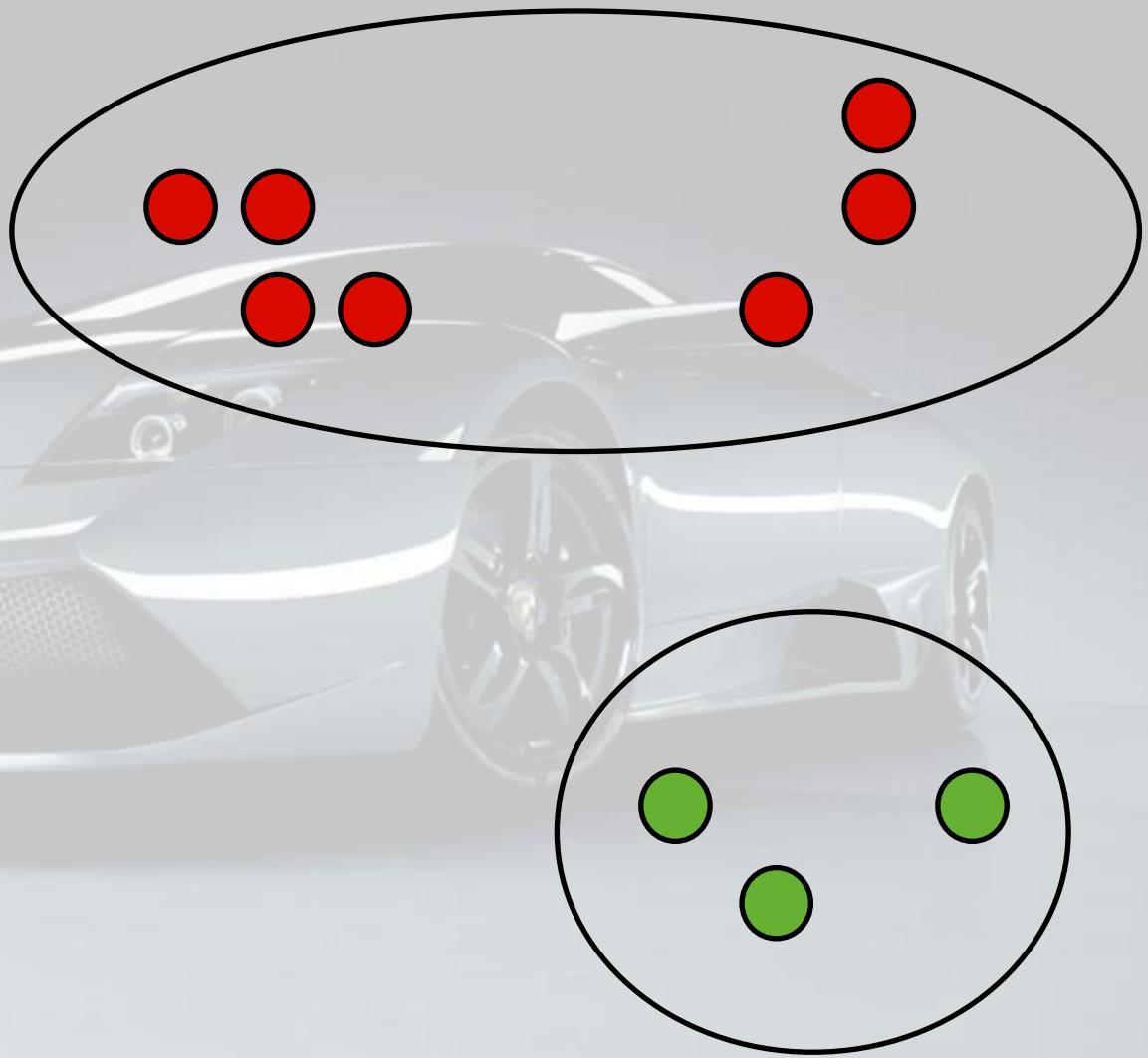
Approach - Clustering

-1.00	0.75
-0.75	0.75
-0.75	0.50
-0.50	0.50
0.75	1.00
0.75	0.75
0.50	0.50
0.25	-0.75
0.50	-1.00



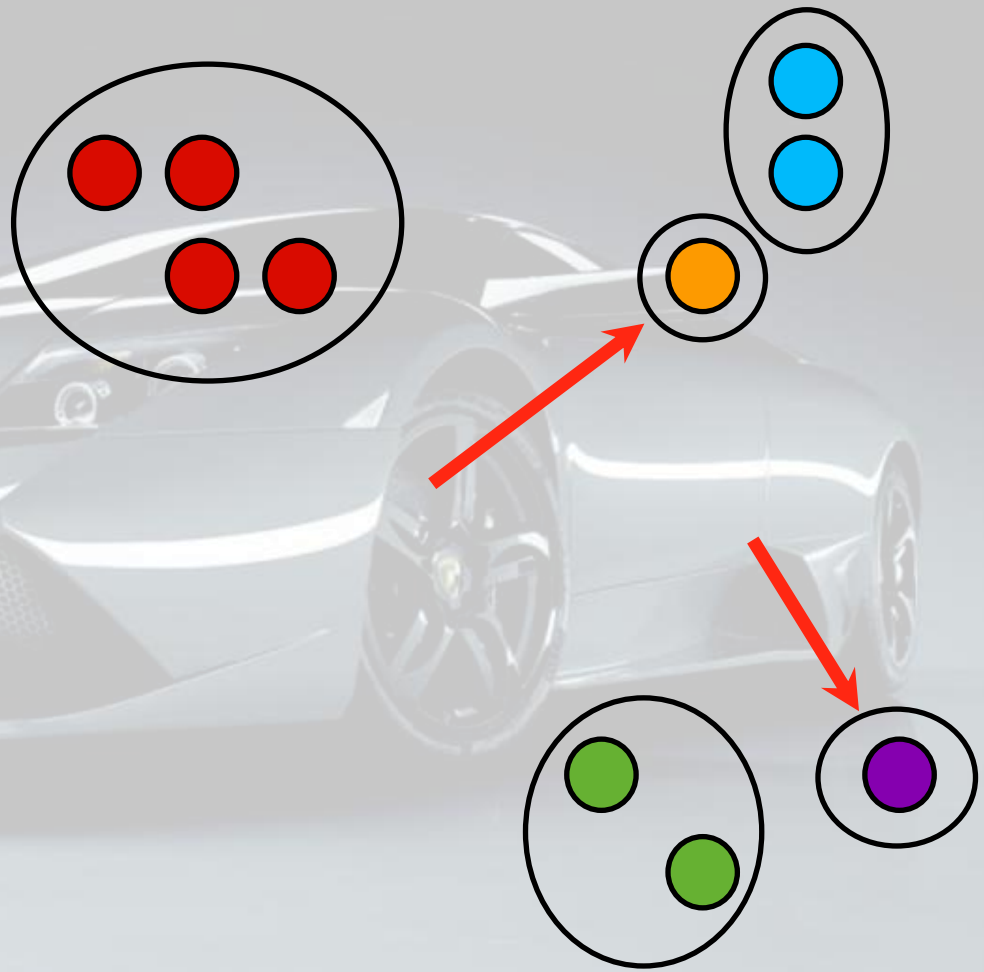
Approach - Clustering

-1.00	0.75
-0.75	0.75
-0.75	0.50
-0.50	0.50
0.75	1.00
0.75	0.75
0.50	0.50
0.25	-0.75
0.50	-1.00



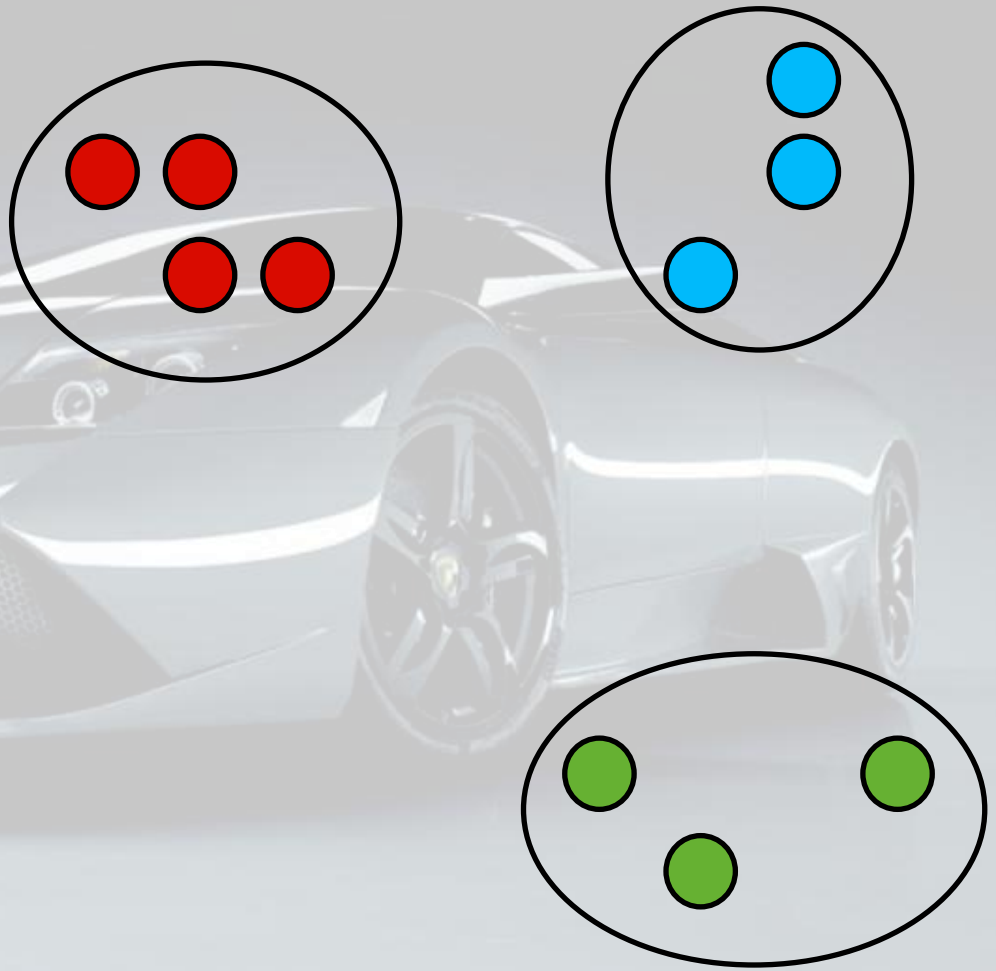
Approach - Clustering

-1.00	0.75
-0.75	0.75
-0.75	0.50
-0.50	0.50
0.75	1.00
0.75	0.75
0.50	0.50
0.25	-0.75
0.50	-1.00

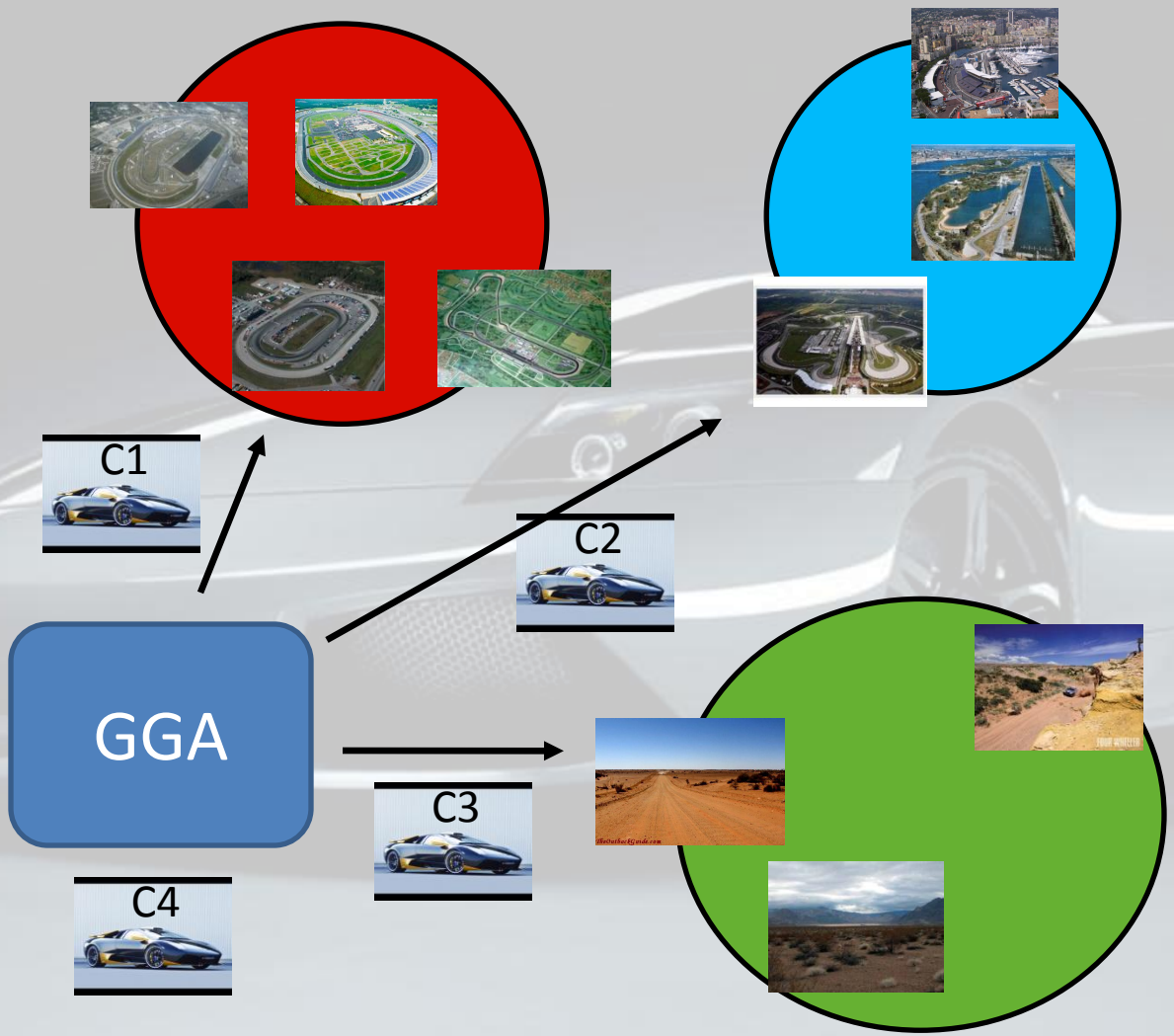


Approach - Clustering

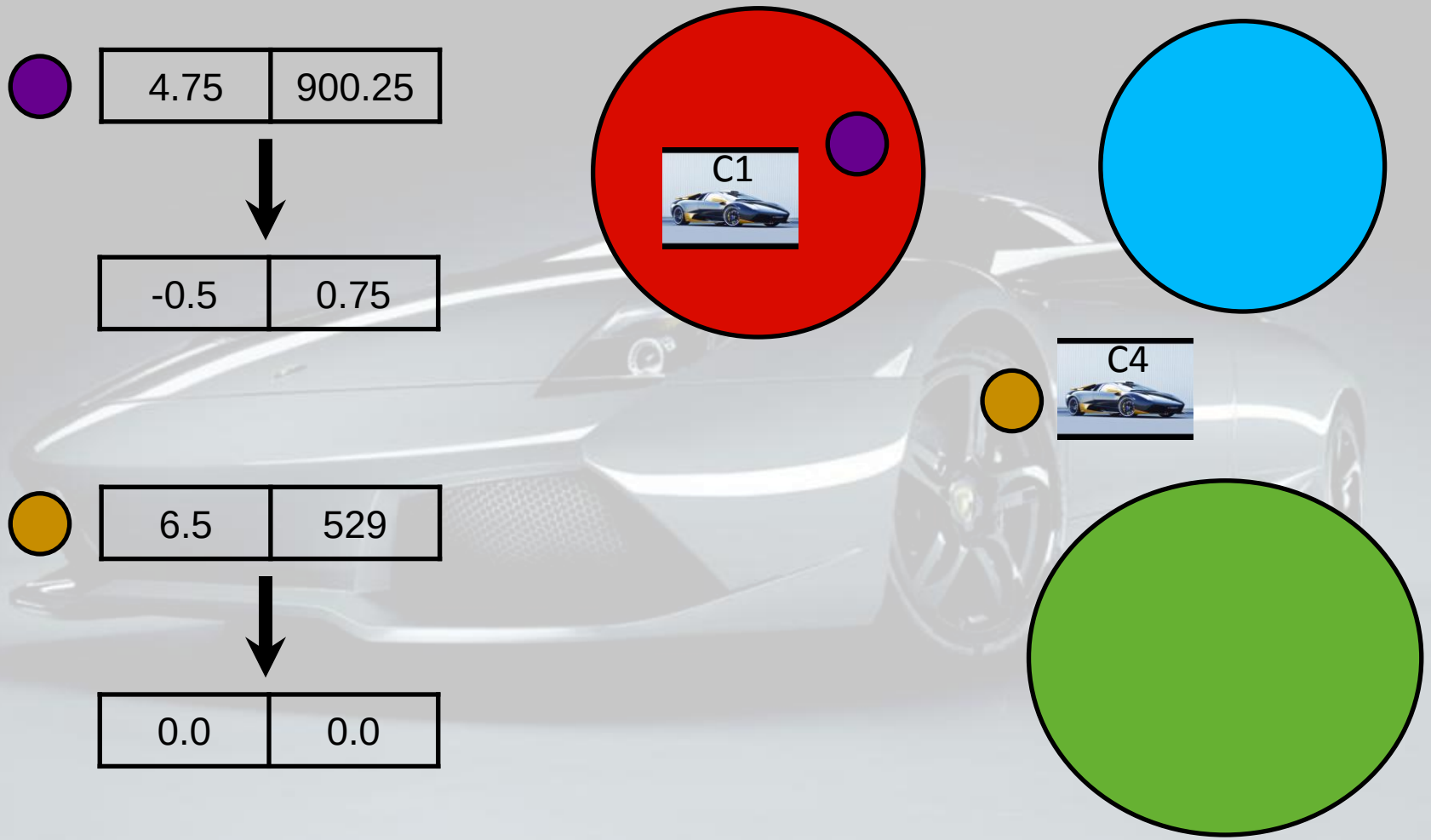
-1.00	0.75
-0.75	0.75
-0.75	0.50
-0.50	0.50
0.75	1.00
0.75	0.75
0.50	0.50
0.25	-0.75
0.50	-1.00



Approach - Tuning



Approach - Runtime



Set Covering Problem

Solver		Avg. Run Time		Geo. Avg		Avg. Slow Down	
		Train	Test	Train	Test	Train	Test
TS (Nysret)	Default	2.79	3.45	2.36	2.60	1.49	1.79
	GGA	2.58	3.40	2.27	2.63	1.35	1.72
	ISAC	**1.99**	**2.04**	**1.96**	**1.97**	**1.0**	**1.0**
Hegel	Default	3.04	3.15	2.52	2.49	2.20	2.03
	GGA	1.58	1.95	**1.23**	**1.33**	1.10	1.15
	ISAC	**1.45**	**1.92**	**1.23**	1.36	**1.0**	**1.0**

Parallel Mixed Integer Programming

Solver		Avg. Run Time		Geo. Avg		Avg. Slow Down	
		Train	Test	Train	Test	Train	Test
Cplex	Default	6.1	7.3	2.5	2.5	2.0	1.9
	GGA	3.6	5.2	1.7	1.8	1.3	1.2
	ISAC	**2.9**	**3.4**	**1.5**	**1.6**	**1.0**	**1.0**

SAT

Solver		Avg. Run Time		Geo. Avg		Avg. Slow Down	
		Train	Test	Train	Test	Train	Test
SAPS	Default	79.7	77.4	0.9	0.9	292.5	274.1
	GGA	14.6	14.6	0.2	0.2	5.5	4.7
	ISAC	**4.0**	**5.0**	**0.1**	**0.1**	**1.0**	**1.0**

SATenstein - BM

Solver	Avg. Run Time		Ave. Par10	
	Train	Test	Train	Test
FACT	4.25	26.0	268	220
Hydra	-	1.43	-	1.43
ISAC	**1.48**	**1.27**	**1.78**	**1.27**

SATenstein - INDU

Solver	Avg. Run Time		Ave. Par10	
	Train	Test	Train	Test
CMBC	5.5	5.35	6.4	5.35
Hydra	-	5.11	-	5.11
ISAC	**2.99**	**2.97**	**2.99**	**2.97**

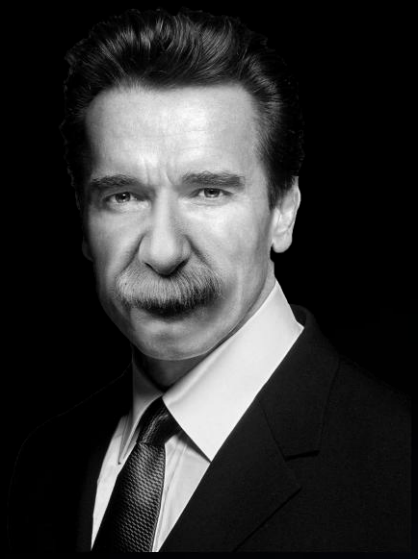
Limitations

- How far from optimal are we?
- Variance in tuning quality?
- How does configuration quality scale with computation time?
- How does final robustness scale with number of training instances?
- How to improve feature distance metric?

More Compute Power!

Conclusions

- Automatic Algorithm Tuning has enormous potential
- State-of-the-art Instance-oblivious Tuning: GGA
  - Population-based approach
  - Gender-separation helps for problems with high evaluations times
  - Inherently parallel
  - Winners determine runtime
- State-of-the-art Algorithm Portfolios: CP-Hydra, 3S
  - Non-model based
  - Exploit scheduling
- State-of-the-art Instance-specific Tuning : ISAC
  - No interpolation
  - Parameters need to work well *together*
  - Offers significant potential over instance-oblivious tuners



THANKS!
QUESTIONS?