Simulation Optimisation

Tutorial

Prof Juergen Branke

GECCO '19 Companion, Prague, Czech Republic
 978-1-4503-6748-6/19/07.$15.00
https://doi.org/10.1145/3319619.3323385

We live in a complex world

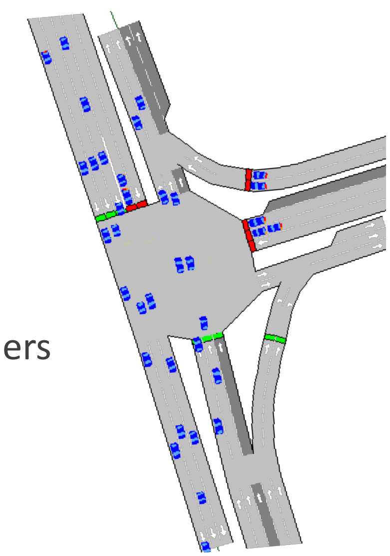
- Large number of interacting elements
- Emergence
- Can not be understood by analysis of components
- Simulation can capture emergent phenomena

Instructor

Juergen Branke is Professor of Operational Research and Systems of Warwick Business School, University of Warwick, UK. He is Area Editor for the Journal of Heuristics and the Journal on Multi Criteria Decision Analysis, and Associate Editor for IEEE Transaction on Evolutionary Computation, and the Evolutionary Computation Journal. His research interests include metaheuristics and Bayesian optimization, multiobjective optimization and decision making, optimization in the presence of uncertainty, and simulation-based optimization. He has published over 180 peer-reviewed papers in international journals and conferences. His e-mail address is Juergen.Branke@wbs.ac.uk

Example: Traffic

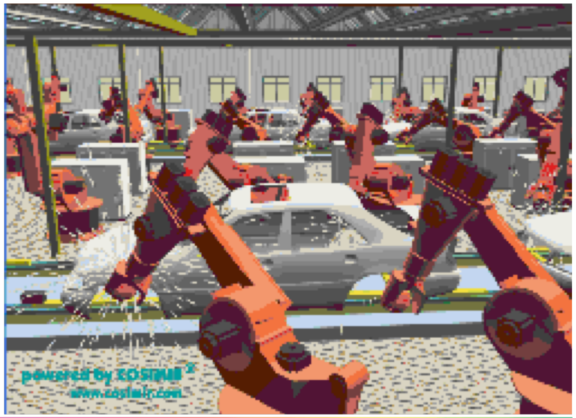
- Street networks
- Reactive traffic light controllers



Example: Manufacturing

Simulate machine breakdowns, stochastic processing times, complex scheduling rules,

etc.



Example: Stock market

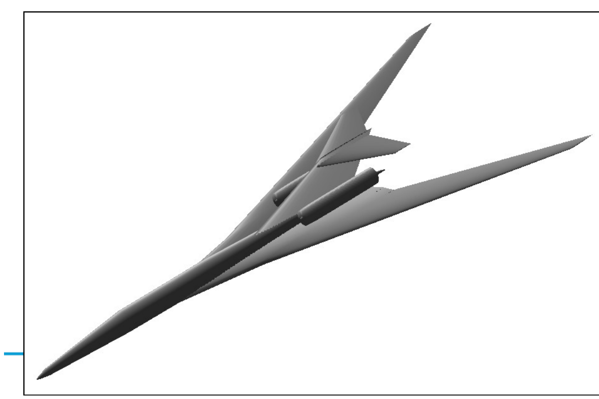
Simulation allows taking into account

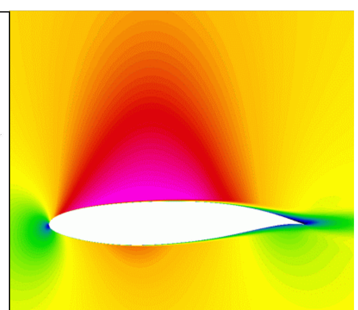
- bounded rationality
- learning agents
- heterogeneous agents
- network effects

Famous:
Santa Fe Stock Market:
Expectations of learning agents lead to technical trading

Example: Engineering

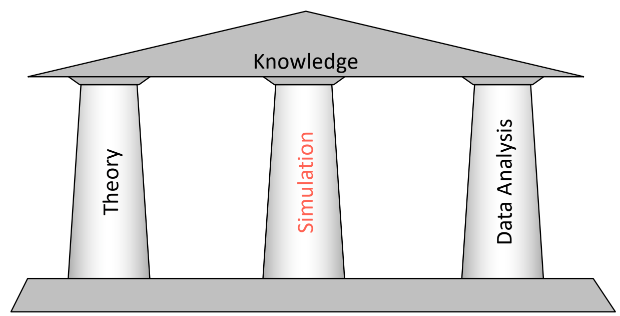
Simulation can replace physical testing





Simulation as knowledge generation tool

Great tool to understand and analyse complex real-world systems!



The next step: Simulation optimisation

- Automatically search vast spaces of parameter settings to find “optimal” settings

- Model calibration
- Automated design and optimisation of complex systems

Challenges

1. Simulations are mostly black boxes 
2. Simulations are computationally expensive
3. Simulations are often stochastic

Simulation optimisation examples

- Traffic: Optimise traffic light controller
- Manufacturing: Find optimal dispatching rules
- Engineering: Find optimal wing design
- Finance: Find better investment strategies

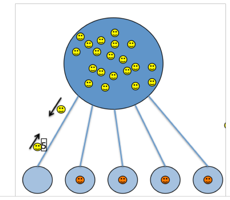
Outline

- Strategies to deal with expensive evaluations
  - Parallelisation
  - Surrogate models
- Strategies to deal with noise
  - Selecting the best system
  - Simulation optimisation
- Applications
  - Design of traffic light controller
  - Design of dispatching rules
  - Design of caching strategies

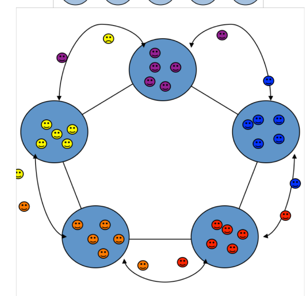
Dealing with expensive evaluations

Parallellisation

a. Parallel evaluations
   - Multi-core/Multi-processor
   - Graphics Processing Units



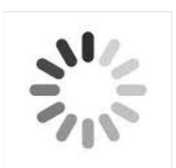
b. Parallel selection
   - Grid/cloud computing



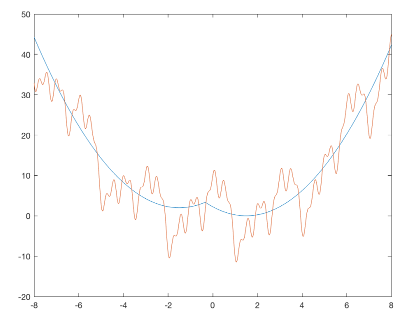
e.g. [Nedjah et al. 2006]

How long can you wait?

- EAs typically require 100,000 function evaluations
- If every simulation takes 1 minute…
- … this is 70 days runtime!



Surrogate models



- "Substitute"
- Often also called "Metamodel"
- Much cheaper, but not necessarily as accurate
- Replace some of the expensive function evaluations by surrogate-model based evaluations

For survey, see e.g. [Jin 2011]

Where does the surrogate model come from?

Simplified:

- More coarse grained simulation
- Smaller simulation
- Abort simulation early

Learned:

- From data systematically sampled from search space
- From data collected during the run

Which solutions to evaluate?

- Promising solutions?
- Representative solutions?
- Solutions where surrogate model is uncertain?
- Solutions that improve accuracy of surrogate model?
- Fixed or flexible budget?

Metamodel design questions

- Type of model, or ensemble
  - Linear/quadratic regression, Gaussian Process, Artificial Neural Network, etc.
- Training data
  - Global vs. local models
- Which individuals to evaluate based on metamodel, which on full model

Learning vs. optimisation

From an optimisation point of view, we want to

- Fully evaluate the best solutions
- Fully evaluate where we are most uncertain
- Ensure the selection works accurately

From a modelling point of view, we want to

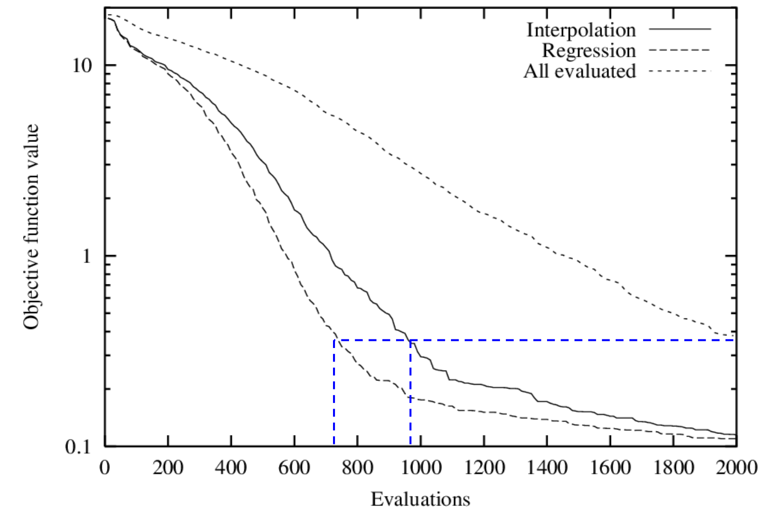
- Evaluate where we can most improve model accuracy
- Evaluate where we are most uncertain

- EAs evaluate many solutions in promising areas, so these areas can be modelled accurately
- Model does not need to accurately predict fitness, only accurately predict ranking

Most typical uses of metamodels

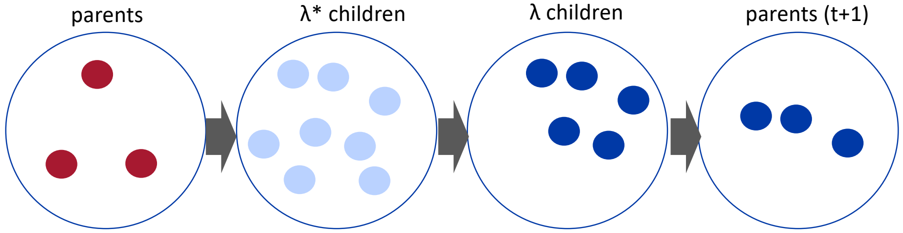
- Pre-selection
- Locally optimise each solution

Benefit of pre-selection [Branke&Schmidt 2004]



Pre-selection

- Generate an abundance of children
- Pre-select λ children based on metamodel
- Fully evaluate pre-selected children



Trust region method

For each individual

Repeat at most k times, or until no better solution found

Build local surrogate model

Perform local search on surrogate model within Trust region

Evaluate best found solution

Replace individual with best found solution if better

Adapt Trust region

Multi-fidelity optimisation

- Sometimes, multiple surrogate models with different trade off between accuracy and running time
- Use fast, rough models to approximate good region
- Use slower, more accurate models to refine the best solution

Efficient Global Optimisation (EGO)

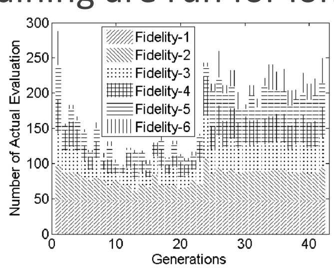
[Jones, Schonlau, Welch 1998]

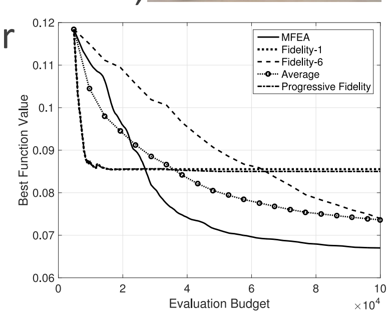
- Fit a Gaussian Process (GP) to data
- Response model provides information about
  - expected value
  - uncertainty
- Use response model to determine next data point (replaces genetic operators)
- Expected improvement makes explicit trade-off between exploration and exploitation

Use of partially converged simulation

[Branke et al. 2017]

- Simulation can be stopped, and later continued
- Evaluate all solutions using short runs
- Clearly good solutions survive, clearly poor solutions are discarded, remaining are run for longer





Efficient Global Optimisation (EGO)

- Fit a Gaussian Process (GP) to data

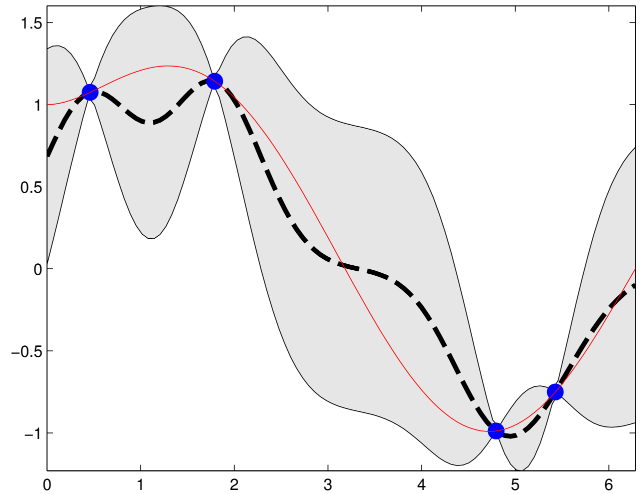
$$f(\vec{x}) \sim GP\big(m(\vec{x}), K(\vec{x}, \vec{x}')\big)$$

where $m(\vec{x}) = 0$

$$\vec{K} = \begin{bmatrix} k(\vec{x}_1, \vec{x}_1) & \cdots & k(\vec{x}_1, \vec{x}_n) \\ \vdots & \ddots & \vdots \\ k(\vec{x}_n, \vec{x}_1) & \cdots & k(\vec{x}_n, \vec{x}_n) \end{bmatrix}$$

$$\underbrace{k(\vec{x}, \vec{x}')}_{\text{kernel}} = \underbrace{\sigma_f^2}_{\text{max cov}} \; exp\left(- \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{2 \underbrace{\ell_d^2}_{\text{length scale}}} \right) + \underbrace{\sigma_n^2 \delta(\vec{x}, \vec{x}')}_{\text{measurement noise}}$$

Example: GP in 1 dimension



EGO algorithm

Take initial n_0 samples

Build GP model

WHILE stopping criterion not met DO

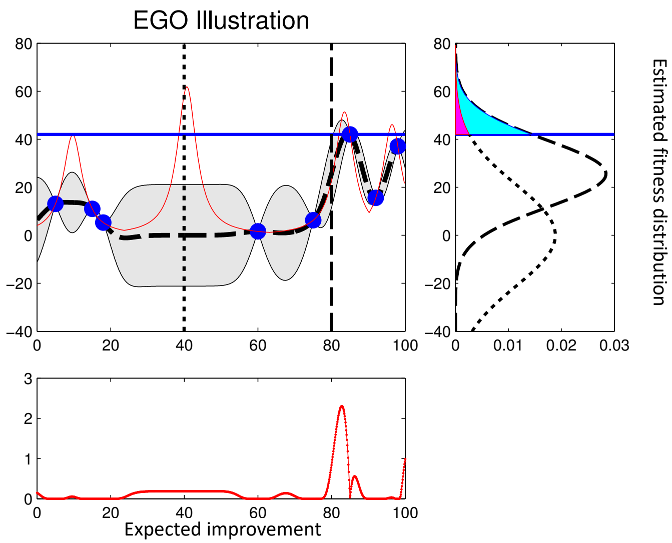
Estimate hyperparameters using maximum likelihood estimation

Take additional sample at position with maximum EI

Update GP model

Return best found solution

Max expected improvement principle



Outline

- Strategies to deal with expensive evaluations
  - Parallelisation
  - Surrogate models
- Strategies to deal with noise
  - Selecting the best system
  - Simulation optimisation
- Applications
  - Design of traffic light controller
  - Design of dispatching rules

Selecting the Best System

Standard: Equal allocation

- Sample each system *n* times
- Reduces standard error by $\frac{1}{\sqrt{n}}$

Ranking and selection problem

- Select, out of *k* systems, the one with best mean performance
- Let X_{ij} be output of *j*th replication of *i*th system
  $\{X_{ij} : j = 1, 2, \dots\} \overset{i.i.d.}{\sim} \text{Normal}(w_i, \sigma_i^2,) \quad i = 1, \dots, k$
- Sample statistics: $\bar{x}_i$ and $\hat{\sigma}_i^2$ based on n_i observations seen so far
- Order statistics: $\bar{x}_{(1)} \leq \bar{x}_{(2)} \leq \dots \leq \bar{x}_{(k)}$
- Correct selection if selected system (*k*) is the true best system [*k*]

Variance Reduction Techniques

- Try to reduce variance without additional runs, but instead by influencing the settings of the experiments
  - Common Random Numbers
  - Antithetic Variates
  - Control Variates
  - …

Common Random Numbers

Intuition:

- Compare two alternatives under similar conditions
- Keep track of performance differences in identical environments
- The observed differences are more likely attributable to the actual system differences, rather than to the differences in environmental conditions

  Var(X-Y)=Var(X)+Var(Y)-2Cov(X,Y)

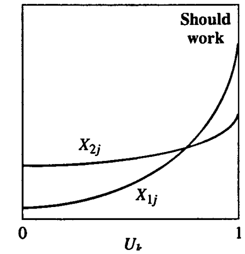
Hope: Cov(X,Y)>0

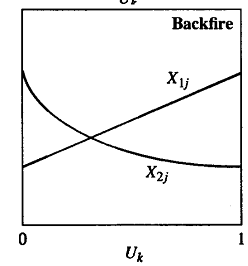
Common Random Numbers

(from: Law&Kelton, 2000)

- CRN works only if random numbers influence performance in the same way
- Can backfire, if that is not the case (rare)

-> Run some initial experiments with and without CRN to test the influence of CRN





Common Random Numbers

How to generate similar conditions

- Conditions are influenced by random number generator
- But: random number generator cannot produce random numbers
- Note: random numbers are used in different contexts
  - arrival rate of customers
  - processing times
  - action selection by agents
  - etc.
- It is necessary to ensure that the random numbers are used for the same purposes in the simulations of the two systems -> synchronization
- Best way to maintain synchronization: Use separate random number streams to corresponding sources of randomness

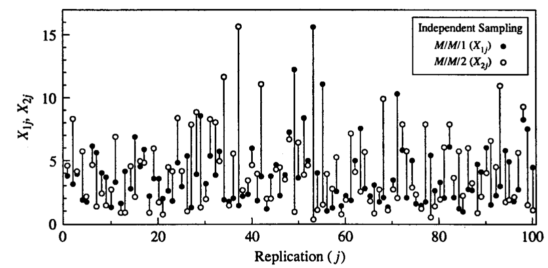
Common Random Numbers

Example:

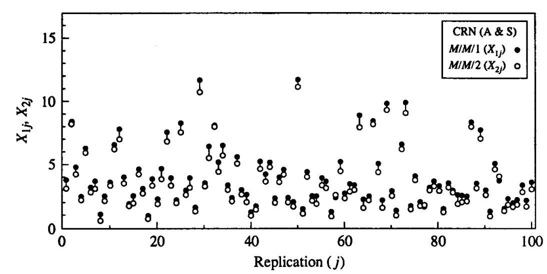
Compare M/M/1 and

M/M/2 production system

(from Law&Kelton,2000)



(a)

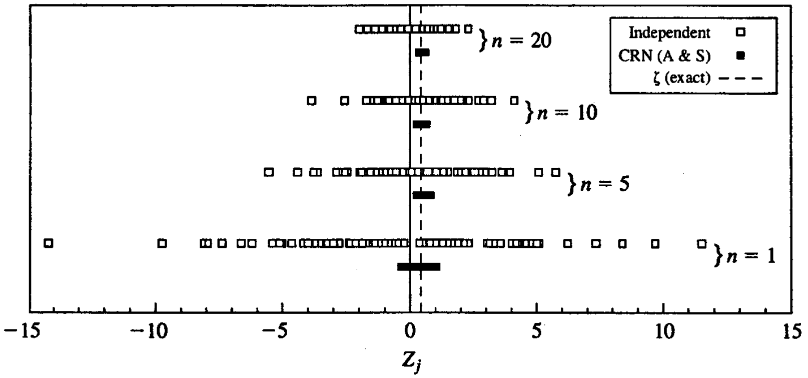


(b)

Comparison of the effect of

- increasing the number of samples (n) and
- using CRN

From Law&Kelton, 2000



But: the effect very much depends on the model properties!

Possible problem: may invalidate (or at least complicate) statistical analysis methods (ranking&selection, ANOVA)

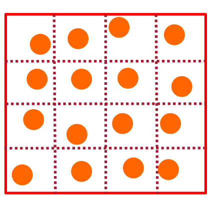
Comparison of m>2 alternatives

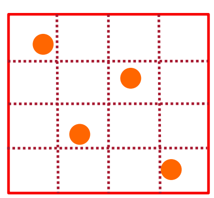
- Allocate samples sequentially
- Maximise the value of information

- Collect more information on promising solutions
- Collect more information where uncertainty is high

Design of experiments

- Stratified sampling
- Latin Hypercube sampling





Performance criteria

- Probability of correct selection (PCS)

$$PCS = P(w_{(k)} = w_{[k]})$$

- Probability of good selection (PGS)

$$PGS = P(w_{(k)} \geq w_{[k]} - \delta)$$

- Expected opportunity cost (EOC)

$$EOC = E(w_{[k]} - w_{(k)})$$

Myopic approach to maximize probability of correct selection

[Chick, Branke, Schmidt: J. of Computing, 2010]

- Assume we can take only one more sample
- If the sample doesn't change selected solution -> information had no value
- PCS: Expected value of information is probability of a change in the index of the individual with the best mean
- EOC: Expected value of information is expected change in the value of the selected individual

Algorithm

Sample each alternative n_0 times

Determine sample statistics $\bar{x}_i$ and σ_i^2 and order statistics $\bar{x}_{(1)} \leq \ldots \leq \bar{x}_{(k)}$

WHILE stopping criterion not reached DO

Take additional sample of system i with maximal EVI

Update sample and order statistics

Pick solution with maximal $\bar{x}_i$

Expected value of information (PCS)

Change of best system if

- system (i) ≠ (k) is evaluated and becomes new best system
- system (k) is evaluated and becomes worse than second best

$$\mathrm{EVI}_{(i)} = \begin{cases} \Phi_{n_{(i)}-1}\left(\dfrac{\bar{x}_{(i)} - \bar{x}_{(k)}}{\sqrt{\frac{\hat{\sigma}^2_{(i)}}{n_{(i)}(n_{(i)}+1)}}}\right) & \text{if } (i) \neq (k) \\ \Phi_{n_{(k)}-1}\left(\dfrac{\bar{x}_{(k-1)} - \bar{x}_{(k)}}{\sqrt{\frac{\hat{\sigma}^2_{(k)}}{n_{(k)}(n_{(k)}+1)}}}\right) & \text{if } (i) = (k) \end{cases}$$

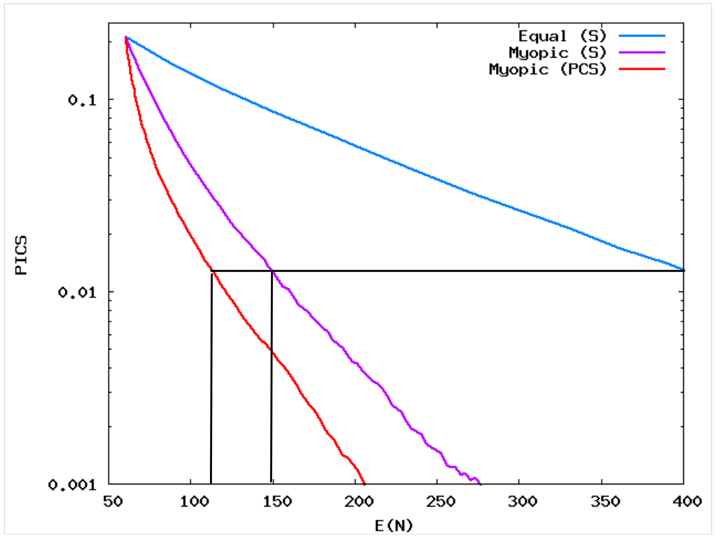
Stopping rule [Branke, Chick, Schmidt, Mngmt Sci, 2007]

- So far: Fixed budget
- Now: Estimate Probability of Correct Selection (PCS)

$$\begin{aligned} \mathrm{PCS}_{\mathrm{Bayes}} &= \Pr(W_{(k)} \geq \max_{j \neq (k)} W_{(j)}) \mid \Xi\} \\ &\geq \prod_{j:(j)\neq(k)} \Pr(W_{(k)} > W_{(j)}) \mid \Xi\} \\ &\approx \prod_{j:(j)\neq(k)} \Phi_{\nu_{(j)(k)}}(d^*_{jk}) \end{aligned}$$

with $d^*_{jk} = (\bar{x}_{(k)} - \bar{x}_{(j)}) \left(\dfrac{\hat{\sigma}^2_{(k)}}{n_{(k)}} + \dfrac{\hat{\sigma}^2_{(j)}}{n_{(j)}} \right)^{-1/2}$

Empirical evaluation (find best out of 10 systems)



Simulation Optimisation

Similar approaches

- Optimal Computing Budget Allocation (OCBA)
  - Asymptotic assumption [Chen&Lee 2011]
- Racing [Birattari et al. 2010]
  - In each iteration, allocate one sample to each alternative "still in the race"
  - F-test to detect whether there is significant difference
  - Eliminate alternatives that are significantly worse than best alternative
  - Stop when budget has been used up or only one alternative is left
  - Version that runs with fixed budget [Branke&Elomari 2012]

If the number of alternatives is large

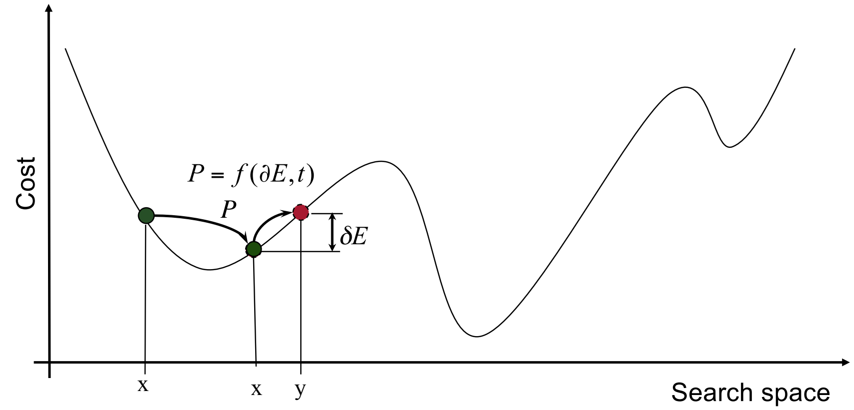
- If the number of possible alternatives is large, it is no longer possible to evaluate each alternative a few times
- We need an optimization heuristic
- Typical:
  Simulated annealing, evolutionary algorithm
- If computational budget is very limited, dimensionality small and variables continuous: Bayesian optimisation

Bayesian Optimisation

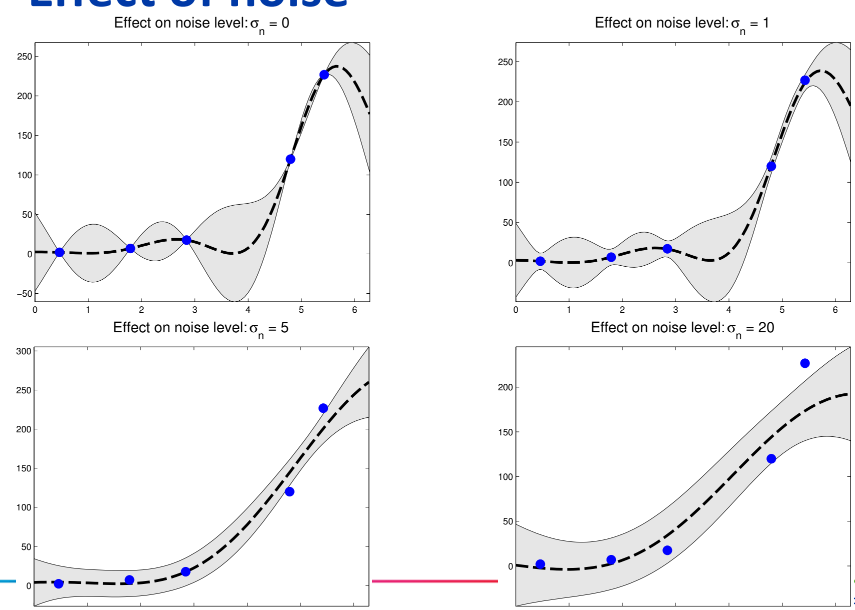
- Extends naturally to the noisy case
- Sequential Parameter Optimisation [Bartz-Beielstein et al. 2005] , Stochastic Kriging Optimization [Huang et al. 2006], Knowledge Gradient [Frazier et al. 2009]

Simulated Annealing

- Stochastic local search
- Inspired by physical annealing processes

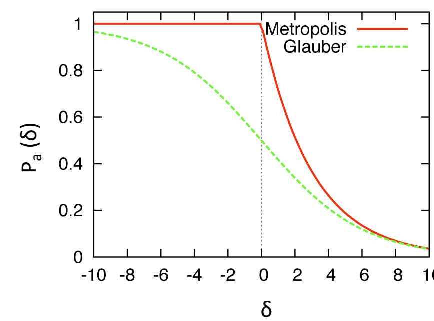


Effect of noise



Simulated Annealing

- Acceptance of solution is probabilistic and depends on quality difference δ and temperature *T*

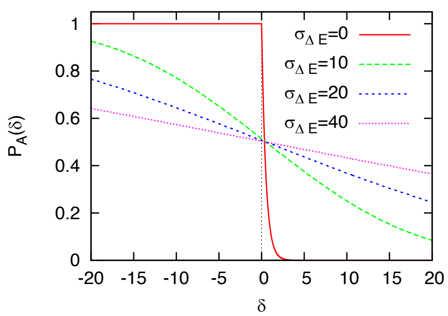


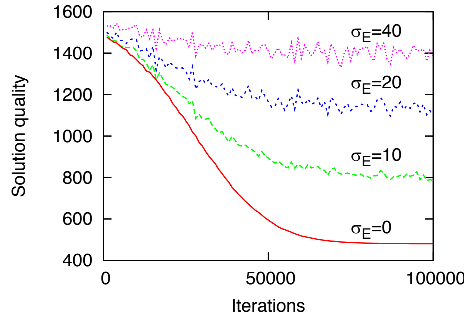
$$\frac{P_a(\delta)}{P_a(-\delta)} = e^{-\delta/T}$$

$$P_a^{Metropolis}(\delta) = \begin{cases} 1 & : \quad \delta \leq 0 \\ e^{-\delta/T} & : \quad \delta > 0 \end{cases}$$

$$P_a^{Glauber}(\delta) = \frac{1}{1 + e^{\delta/T}}$$

Effect of noise





Simulated Annealing in Noisy Environments (SANE) [Branke et al. 2008]

Idea:

- Always accept seemingly better solution
- Number of samples depends on temperature and probability to accept worse solution
- Keep sampling until the probability to erroneously select the worse solution is smaller than the acceptance probability for the worse solution

Using the noise

Noise in SA/EA

Where?

- Selection
- Mutation
- Replacement

What for?

- Get out of local optima
- Explore search space

How?

- Artificial, pseudo-random

Noise in real-world problems

generally stochastic environment

What from?

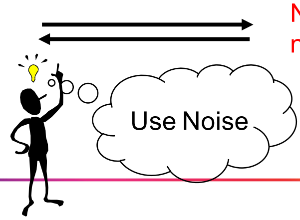
- Simulation based evaluation
- Measure errors
- Uncertainty

How?

- "Naturally"

Noise desired

Noise not desired



Simulated Annealing in Noisy Environments (SANE) [Branke et al 2008]

$n = n_0 = 1$

$\boxed{\text{Draw } n_0 = 1 \text{ sample from } \Delta E \text{ and estimate } \delta \text{ by } \hat{\delta}}$

$\boxed{P_{err}(\hat{\delta}) = \Phi\left(\frac{-|\hat{\delta}|\sqrt{n}}{\sigma_{\Delta E}}\right)}$

$\boxed{\textbf{while } P_{err} > P_a^{Glauber}(|\hat{\delta}|) \textbf{ do}}$ while estimated error probability is greater than Glauber's probability of picking worse

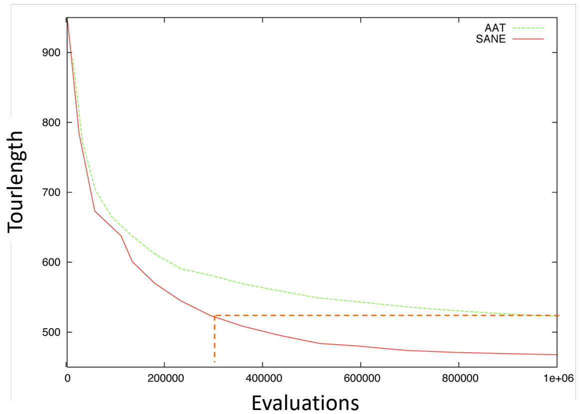
$\boxed{\text{Draw another sample } (n \leftarrow n+1)}$
$\boxed{\text{Update } \hat{\delta} \text{ and } P_{err}}$

end while

$\boxed{\text{Accept better solution}}$ always pick better solution

Empirical comparison

- TSP with normally distributed noise
- Comparison with Alkhamis et al. (AAT) [Alkhamis, Ahmed, Tuan, 1999]



OSA acceptance rule

- Based on sum of samples taken so far

$$c_n = \sum_{i=1}^{n} \delta_i$$

- Acceptance probability at current stage:

$$A(c_n, c_{n-1}) = \begin{cases} 1 & c_n < -\beta\sigma^2/2 \\ e^{-2(c_n+\beta\sigma^2/2)(c_{n-1}+\beta\sigma^2/2)} & \text{otherwise} \end{cases}$$

- If not accepted, reject if c_n>0
- Continue otherwise

Optimal Stochastic Annealing (OSA)

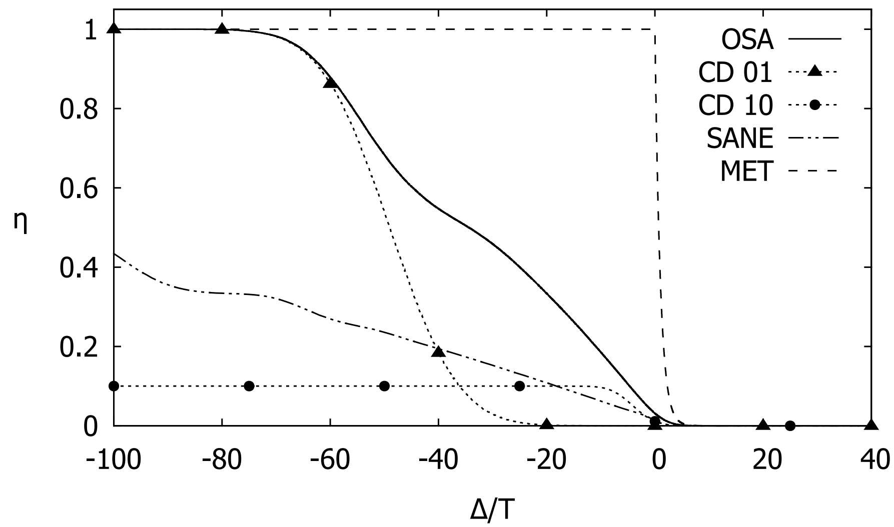
[Ball et al. 2017]

- Assumes known Gaussian noise
- Uses sequential sampling
- At every stage, decision to accept, reject or continue
- Acceptance criterion modified to maintain detailed balance
- Acceptance criterion has optimal efficiency (acceptance probability per sample)

Benchmark algorithms

- SANE [Branke et al. 2007]
  - Sequential sampling and adjusted acceptance criterion
  - Current state-of-the-art, shown to outperform several other methods
- CD1 [Ceperley&Dewing 1999]
  - Adjusted acceptance criterion, obeys detailed balance
- CD10[Ceperley&Dewing 1999]
  - As CD1, but with 10 samples per move decision

Efficiency (σ/T=10)



Evolutionary algorithm

INITIALIZE population
(set of solutions)
EVALUATE Individuals
according to goal ("*fitness*")

REPEAT

SELECT parents

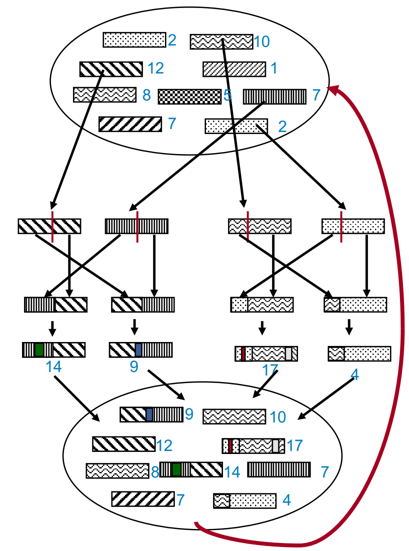
RECOMBINE parents (CROSSOVER)

MUTATE offspring

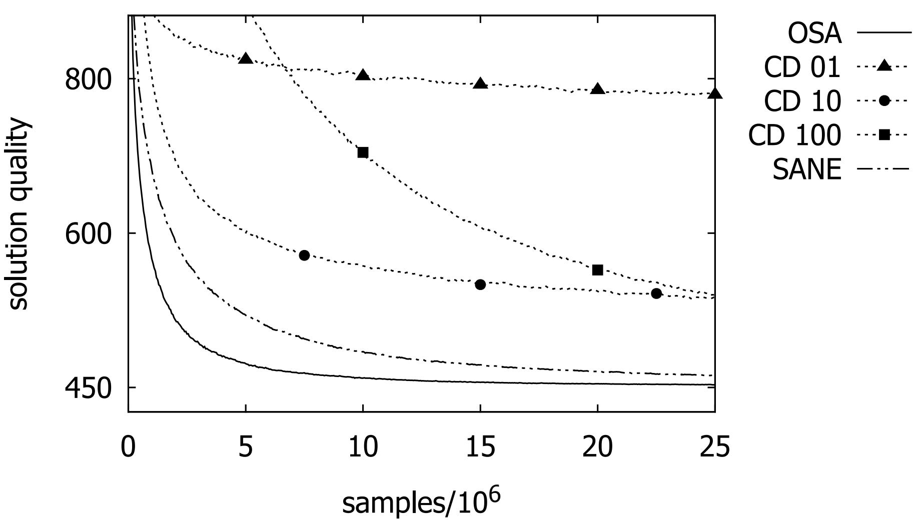
EVALUATE offspring

FORM next population

UNTIL termination-condition



Optimization performance (TSP, σ^2=3200)



Populations are robust to noise

- Implicit averaging over the neighbourhood
- With infinite populations, fitness proportional selection is not affected by noise [Miller & Goldberg 1996]
- Theory for optimal population sizes in simplified cases [Arnold&Beyer 2000]
- Black-box Optimization Benchmark competitions show advantages of EAs in noisy environments

Explicit averaging

- Reduce noise by factor sqrt(n)

Change sample size over the run

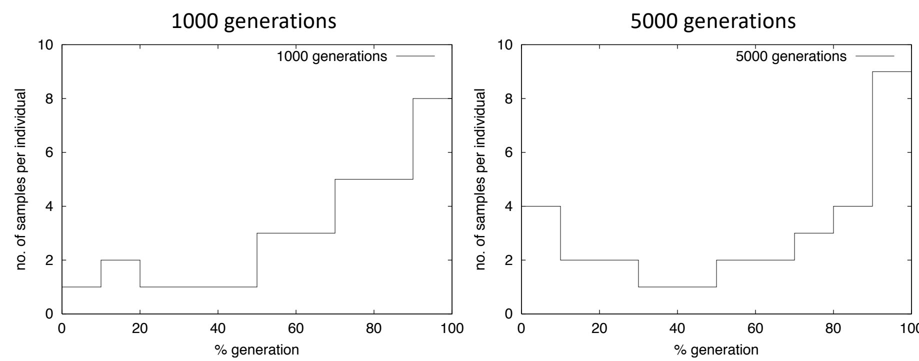
- Increase sample size over the run [Aizawah&Wah 1993]
- Optimise distribution of samples over the run [Branke 2001]
- Clean up after optimisation [Boesel et al. 03]

CRN and Evolutionary Algorithms

[Branke 2001]

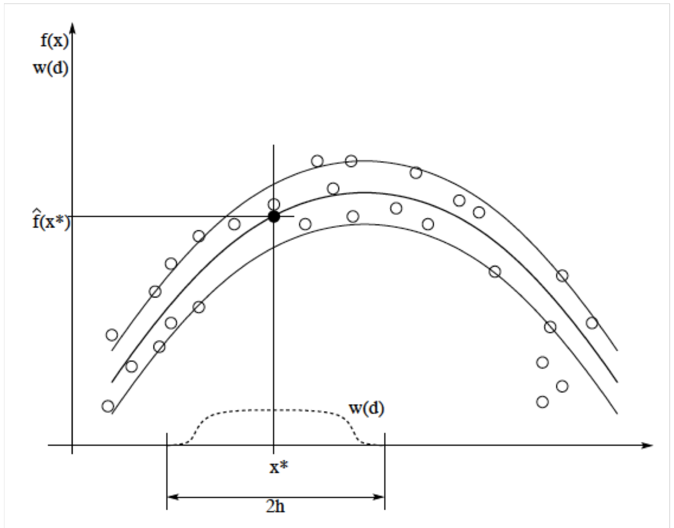
- Use CRN for all individuals to be compared within a generation
  - may drastically improve probability of correct ranking
  - risk of optimizing for one random seed
- Change random number seeds from generation to generation
  - Only individuals that work on a wide range of scenarios will survive for a long time
- Re-evaluate elite individuals
  - A "lucky" individual should be prevented from surviving forever

Optimal distribution of samples over run [Branke 2001]



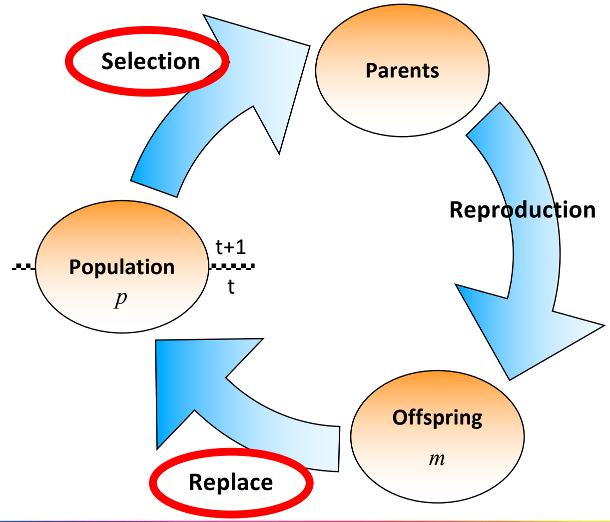
Use metamodels – average over space

[Branke & Schmidt 2001]

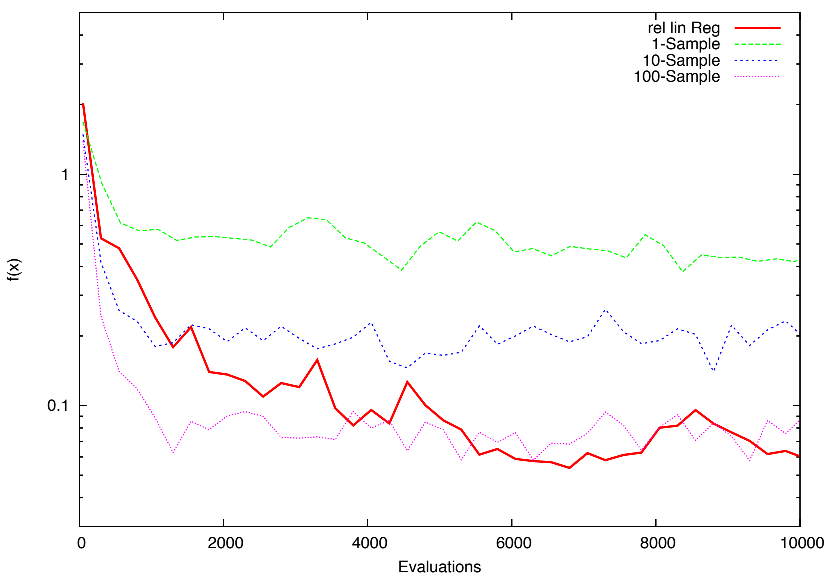


Integrating Ranking&Selection

[Schmidt et al. 2006]



Benefit



The relevant comparisons

- Steady-State-EA with 2-Tournament
  Population size: 9, offspring: 1
  - Replacement: Worst individual
  - Stopping criterion: Best individual
  - Selection: Best out of {3, 7} and {2, 5}
- (5,10)-Evolution strategy
  Population size:5, offspring: 10
  - Replacement: 5 best individuals
  - Stopping criterion : Best individual

Observed ranking

≥	1	2	3	4	5	6	7	8	9	10
1										
2	x									
3	x									
4	x									
5	x									
6	x	x	x	x	x					
7	x	x	x	x	x					
8	x	x	x	x	x					
9	x	x	x	x	x					
10	x	x	x	x	x					

Observed ranking

Observed ranking

≥	1	2	3	4	5	6	7	8	9	10
1										
2	x									
3	x									
4	x									
5	x	x								
6	x									
7	x		x							
8	x									
9	x									
10	x	x	x	x	x	x	x	x	x	

Observed ranking

$$PCS^{EA}_{Step,\delta^*} = \prod_{(i,j)\in C} \Phi_{\nu_{(k)}}\left((d_{ij}+\delta^*)/s_{ij}\right)$$

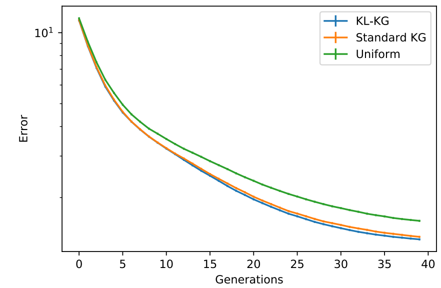
Integrating OCBA and EA

Procedure OCBA^{EA}

1. Evaluate each new individual n_0 times. Estimate the ranks
2. Determine set of relevant comparisons C
3. WHILE evidence is not sufficient
   a) allocate new sample to individual according to modified OCBA rule
   b) if ranks have changed, update C

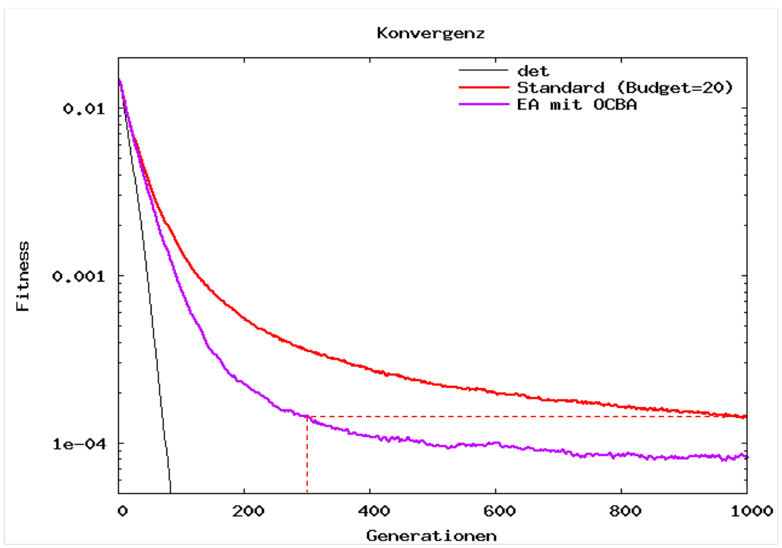
Integrating KG and CMA-ES

- CMA-ES only needs to identify top μ individuals
- Uses this to adapt the mutation step size
- Which individual, if re-evaluated, has biggest potential impact on resulting mutation distribution?
- See paper at this GECCO



(d) Ackley Function

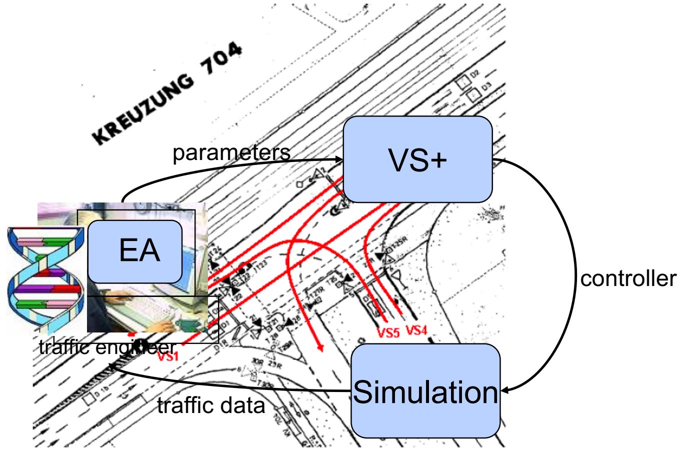
Benefits over the run



Outline

- Strategies to deal with expensive evaluations
  - Parallelisation
  - Surrogate models
- Strategies to deal with noise
  - Selecting the best system
  - Simulation optimisation
- Applications
  - Design of traffic light controller
  - Design of dispatching rules
  - Design of caching strategies

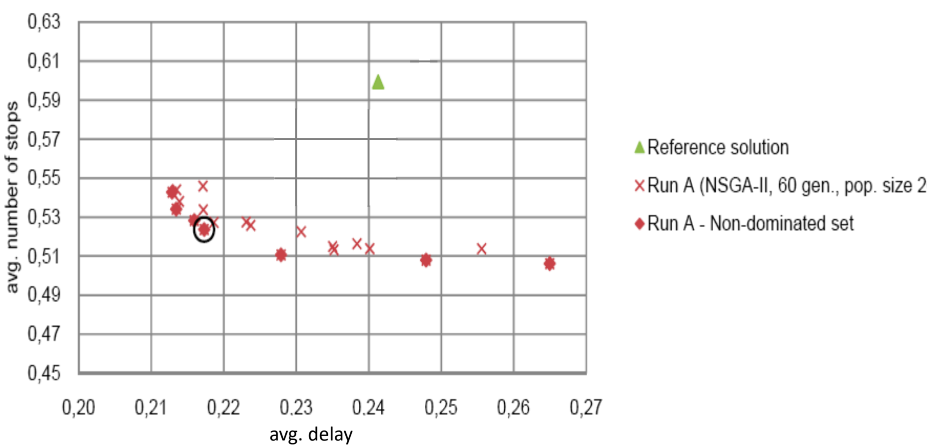
Example: Optimisation of a traffic light controller



Example 2: Evolution of Job Shop Dispatching Rules [Pickardt et al. 2012]

- Complex wafer manufacturing system
  - 31 work centres (35 machines)
  - 10 batching machines, 2 machines with setup times
  - 7 different products
  - 20-100 operations per job (cycles)

Results



EA finds better solutions than traffic engineer

Dispatching rule based scheduling

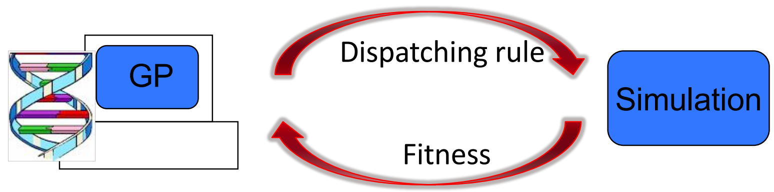
- Determine job priorities based on job and machine attributes
- Whenever a machine becomes idle, process the job with highest priority next
- Popular examples: FIFO, SPT, EDD, CoverT

Advantages:

- Always take latest information into account
- Easy to implement and to compute

Automatic generation of dispatching rules

- Genetic Programming can generate Lisp expressions
- Evaluation of a dispatching rule via stochastic simulation



Results

- Rule of length 9: w/max(L,P)-s+b
- Rule of length 98:

$$\mathrm{ifte}(\max(1,r)-\max(1,r,L),w,b)*b*\max(r/L+\max(-\mathrm{ifte}(b-L,w,b)+s+b,S+b*\\ \mathrm{ifte}(\max(1,r)-\max(L,d),w,b)-s-\max(1,r,L)+\max(1,r)+1)*\mathrm{ifte}(b-L,w,b)-s,S+\\ b*\mathrm{ifte}(\max(1,r)-L,w,b)*(2*r/L-s)+r/L-s+1)$$

Terminals

- Processing time
- Processing time on next machine
- Number of operations remaining
- Remaining processing time
- Work in next queue
- Time in queue
- Time in system
- Slack
- Time until deadline
- Weight
- Setup time
- Number of compatible jobs for batching

Results (2)

Comparison with best rules from literature

Util 93.8%; Product mix 30/70

Rule	WeightedTardiness
ATCS/MBS(5)	2336
GP98	782

Util 85%; Product mix 30/70

Rule	WeightedTardiness
ATCS/MBS(4)	451
GP98	47

Util 85%; Product mix 70/30

Rule	WeightedTardiness
WMOD/MBS(1)	216
GP98	51

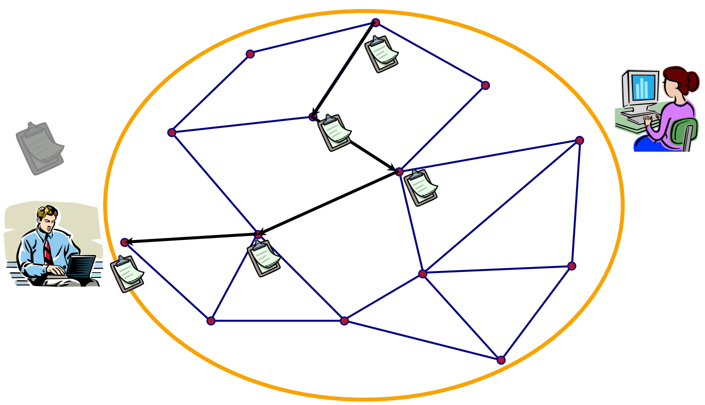
Util 93.8%; Product mix 70/30

Rule	WeightedTardiness
WMOD/MBS(3)	1245
GP98	206

Example 3: Evolving Caching Strategies

The WWW: A huge distributed database **[Branke et al. 2007]**

Documents are relayed by a sequence of routers



Potential solution: Caching

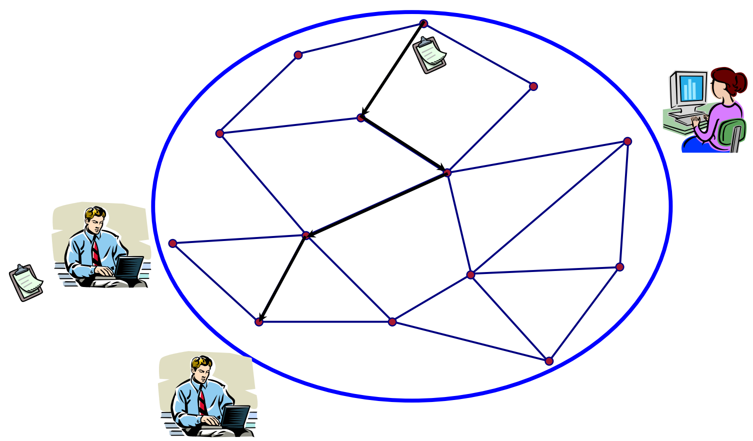
- Storing replicas of frequently requested documents at nearby nodes
- Possible because requested servers and documents have powerlaw-distribution
- Common on browser level and proxy level
- New idea: En-route web caching

Goals:

- Reduce Internet traffic
- Reduce load on highly requested servers
- Reduce latencies

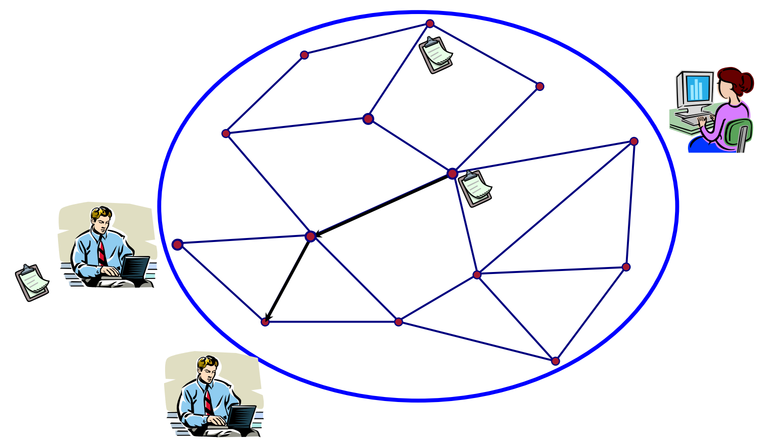
The same document is sent many times

Problems: congestion, delays, timeouts, …



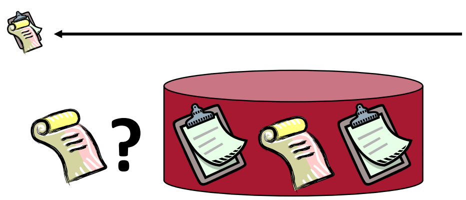
With En-Route Caching

Second request can be serviced from nearby replica



En-Route Caching

- All nodes/routers involved in relaying an object have an opportunity to keep a cached copy
- Potential for huge resource savings
- Problem: limited memory

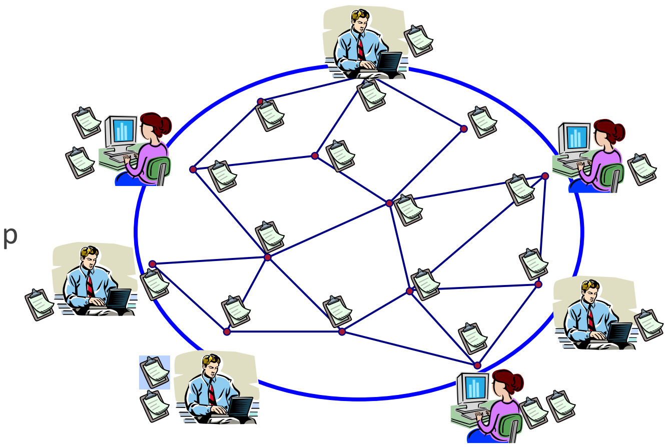


Caching Policy = Decision Rule

- Question: Which documents to delete?

Cache Symmetry Problem

- All nodes use identical rule
- All may end up with identical cache



➡ **Caches should complement each other**

➡ **Coordination necessary**

Challenges

- Large, dynamically changing distributed system
- No global information
- No global authority
- How good a strategy is depends on what neighbors do
- Symmetry problem
- How can global efficiency and coordination emerge from local caching rules?

State of the art:

- LRU (default rule, ignores network aspects)
- RANDOM (often better than LRU, ignores everything)
- GDSF [Cherkasova 2001]

$$priority = \frac{access\ count \cdot distance}{size} + aging\ factor$$

Goal:

Automated design of better en-route caching strategies

Use evolutionary algorithms to explore the space of caching rules

Evolving caching strategies

Main challenges:

- Definition of search space
  - Identify relevant document attributes
  - Use GP to form priority rule
- How to evaluate a caching strategy
  - Only simulation possible due to emergent behavior
  - Requires parallelisation

Network Simulator

1. Create network topology
   - Find paths from all to all
2. Create random set of objects
   - Random size
   - Power-law distributed set of demands
   - Distributed among hosts
3. Poisson process for each host
   - Generate requests for documents
4. Routing
   - Break object in packets, send them along shortest path
5. Bandwidth
   - Each network link is a queue of requests
6. Caching
   - Always send first replica found down request path

GP Evolves Caching Strategies

Priority rules encoded by GP

Inputs = information about the object

a) Time of document creation
b) Document size
c) Access count
d) Time of last access
e) Distance from sender
f) Frequency (no. accesses / second)
g) Random Constant

Functions:

+   *   –   /   sin   cos   exp   iflte

Output = priority

Internet-like Networks (scale-free)

- Internet-like random networks
  - 100 nodes
  - Scale-free topology (Bu & Towsley 02)
- Noisy fitness
  - Different random topologies and request patterns lead to large differences in latency
  - Test in 3 different random nets per generation
  - Change test scenario in every generation
  - Evaluate results on many networks

RUDF

- Often, resulting rules are very complex

```
(* (* b (+ (+ (* b (+ (+ a (- 0.994 b)) (* b (+ (* b e) (exp
(exp e)))))) (* d e)) (exp e))) (iflte (iflte (+ (exp f) (*
(iflte 0.694 f d a) (- b b))) (exp (exp (* (* (+ a f) f) (exp
c)))) (iflte b (* c (- 0.139 (* 1.616 a))) 0.444 (iflte e
(iflte d a 0.601 (- f (- a e))) c f)) b) e 0.507 (exp (* c
(+ (+ (exp d) d) (* a b))))))
```

- One of the runs yielded a short, powerful rule

  lastTimeAccessed ⍰ (distance + accessCount)

- RUDF outperforms all the comparison rules, including GDSF.

Linear Networks

- We can find optimal caching analytically

Node	0	1	2	3	4	5	6	7	8	9
Original	0	1	2	3	4	5	6	7	8	9
Replica	5	6	7	8	9	0	1	2	3	4

- Result: Evolved GP policy yields near-optimal performance, far better than comparison algorithms

Caching strategy	Avg. latency
OPTIMAL	31.58 ± 0.03
BESTGP	31.98 ± 0.06
RUDF	35.80 ± 0.43
GDSF	47.67 ± 1.17
DISTANCE	50.40 ± 1.24
RANDOM	61.65 ± 0.14
LRU	74.77 ± 0.19

Performance on test networks

	Caching Strategy	Ø rank(latency)
30 nodes	RUDF	1.13 ± 0.06
	GDSF	2.80 ± 0.15
	DISTANCE	3.10 ± 0.28
	LRU	3.70 ± 0.16
	RANDOM	4.27 ± 0.14
300 nodes	RUDF	1.03 ± 0.03
	DISTANCE	2.23 ± 0.16
	GDSF	3.20 ± 0.11
	RANDOM	4.20 ± 0.16
	LRU	4.33 ± 0.13

wbs WARWICK BUSINESS SCHOOL

Conclusion and additional resources

Conclusion

- Combining simulation and optimisation is the next step in the design of complex systems
- Metaheuristics hold great promise
- Challenges of runtime and noise can be tackled

Further resources

- The Winter Simulation Conference always has a stream on simulation optimisation
- GECCO tutorial on cloud computing
- Library of simulation optimisation problems http://www.simopt.org
- Available solvers:
  - OptQuest (http://http://www.opttek.com/OptQuest)
  - COMPASS (http://www.iscompass.net)
  - SPOT (https://**cran.r-project.org**/web/packages/**SPOT/**)
  - irace (http://iridia.ulb.ac.be/irace/)

What I have not talked about

- Stochastic Approximation
- Handling of multiple objectives
- Combinatorial problems
- Warm-up period
- Worst-case optimisation

References

- Aizawa, A.N. and Wah, B. W. (1993). Dynamic control of genetic algorithms in a noisy environment. In International Conference on Genetic Algorithms, pp. 48–55
- Arnold, D. V.; Beyer, H.-G. (2000). Efficiency and mutation strength adap- tation of the -ES in a noisy environment. In Parallel Problem Solving from Nature. LNCS 1917, Springer, pp. 39–48.
- Ball, R.; Branke, J.; Meisel, S. (2017) Optimal Sampling for Simulated Annealing Under Noise. INFORMS Journal on Computing 30(1):200-215
- Bartz-Beielstein, T.; Lasarczyk, C.; Preuß, M. (2005) Sequential parameter optimization. In: McKay B, et al (eds) Congress on Evolutionary Computation, IEEE Press, vol 1, pp. 773–780
- Birattari, M.; Yuan, Z.; Balaprakash, P.; Stützle, T. (2010). F-Race and Iterated F-Race: An overview. In: Experimental Methods for the Analysis of Optimization Algorithms, pp. 311-336
- Boesel, J.; Nelson, B. L.; Kim, S.-H. (2003). Using ranking and selection to clean up after simulation optimization. Operations Research 51(5):814-825
- Branke, J. (2001) Reducing the sampling variance when searching for robust solutions, Genetic and Evolutionary Computation Conference, Morgan Kaufmann, pp. 235-242

- Branke, J. (2001). Evolutionary optimization in dynamic environments. Kluwer
- Branke, J.; Chick, S.; Schmidt, C. (2007). Selecting a selection procedure. Management Science 53(12):1916-1932
- Branke, J.; Elomari, J. (2012). Meta-optimization for parameter tuning with a flexible computing budget. Genetic and Evolutionary Computation Conference, ACM, pp. 1245-1252
- Branke, J.; Funes, P.; Thiele, F. (2007). Evolving en-route caching strategies for the Internet. Applied Soft Computing Journal 7(3):890-898
- Branke, J.; Meisel, S.; Schmidt, C.(2008). Simulated annealing in the presence of noise. Journal of Heuristics 14:627-654
- Branke, J.; Asafuddoula, M.; Bhattacharjee, K.; Ray, T. (2017) Efficient Use of Partially Converged Simulations in Evolutionary Optimization, IEEE Transactions on Evolutionary Computation 21(1):52-64
- Chen, H.-C.; Lee, L.-H. (2011). Stochastic simulation optimization: an optimal computing budget allocation. World Scientific
- Chick, S; Branke, J; Schmidt, C.(2010). Sequential sampling to myopically maximize the expected value of information". Informs Journal on Computing 22(1):71-80
- Frazier, P.; Powell, W.; Dayanik, S.: The knowledge gradient policy for correlated normal beliefs. INFORMS Journal on Computing 21(4):599-613

- Fu, M. (2002): Optimization for simulation: Theory vs. practice. Informs Journal on Computing 14(3):192-215
- Fu, M. (ed., 2015): Handbook of Simulation Optimization. Springer
- Hong, L.J.; Nelson, B.L. (2007). Selecting the best system when systems are revealed sequentially. IIE Transactions, 39:723-734
- Huang, D.; Allen, TT.; Notz, WI.; Zeng, N. (2006) Global optimization of stochastic black-box systems via sequential kriging meta-models. Journal of Global Optimization 34(3):441–466
- Jin, Y.; (2011) Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation 1(2):61-70
- Jin, Y.; Branke, J. (2005) Evolutionary optimization in uncertain environments – A survey. IEEE Transactions on Evolutionary Computation 9(3):303-318
- Jones, D. R.; Schonlau, M.; Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13:455-492
- Law, A.; Kelton, W. D. (2001). Simulation Modeling and Analysis. McGraw Hill
- Miller, B. L.; Goldberg, D. E. (1996). Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 4(2):113–131
- Schmidt, C.; Branke, J.; Chick, S. (2006). Integrating techniques from statistical ranking into evolutionary algorithms. Applications of Evolutionary Computation, Springer, LNCS 3907, pp. 753-762