

# Increasing the Efficiency of Physical Optimization Algorithms

Johannes Josef Schneider

Department of Physics  
Johannes Gutenberg University of Mainz

# Overview of Optimization Algorithms

- Exact Algorithms

- Simplex algorithm  
linear cost function, constraints given as inequalities
- Branch&Bound  
intelligent search with upper and lower bounds
- Branch&Cut  
solves problems iteratively by adding further cutting planes or performing branching steps

- Heuristic Algorithms

- Construction Heuristics
  - \* starting from a Tabula Rasa (e.g., Bestinsertion)
  - \* starting from an extended system (e.g., Savings)
- Improvement Heuristics
  - \* Simulated Annealing
  - \* Genetic Algorithms
  - \* Tabu Search
  - \* Guided Local Search
  - \* Ant Colony Optimization

# Introduction in Physical Optimization Algorithms

Classic Algorithm:

Simulated Annealing

Ingredients:

- Boltzmann Distribution (problem = classic physical system)

$$p(\sigma) = \frac{1}{Z} \exp\left(-\frac{\mathcal{H}(\sigma)}{k_B T}\right)$$

partition sum  $Z = \sum_{\sigma} \exp(-\mathcal{H}(\sigma)/(k_B T))$

$p(\sigma)$  = probability to be in state  $\sigma$

- Detailed Balance (strong condition for equilibrium)

$$p(\sigma)W(\sigma \rightarrow \tau) = p(\tau)W(\tau \rightarrow \sigma)$$

$W(\sigma \rightarrow \tau)$  = transition probability from the state  $\sigma$  to the state  $\tau$

$$\frac{W(\sigma \rightarrow \tau)}{W(\tau \rightarrow \sigma)} = \frac{p(\tau)}{p(\sigma)} = \exp\left(-\frac{\Delta\mathcal{H}}{k_B T}\right)$$

Some arbitrariness in the explicit choice of  $W$  remains.

→ Metropolis Criterion

$$W(\sigma \rightarrow \tau) = \min \left\{ 1, \exp \left( -\frac{\Delta \mathcal{H}}{k_B T} \right) \right\} = \begin{cases} \exp \left( -\frac{\Delta \mathcal{H}}{k_B T} \right) & \text{if } \Delta \mathcal{H} \geq 0 \\ 1 & \text{otherwise} \end{cases}$$

or Heatbath Criterion

$$W(\sigma \rightarrow \tau) = \frac{p(\tau)}{p(\sigma) + p(\tau)} = \frac{1}{1 + \exp \left( \frac{\Delta \mathcal{H}}{k_B T} \right)}$$

# Algorithms related to Simulated Annealing

## Threshold Accepting

$$W(\sigma \rightarrow \tau) = \begin{cases} 1 & \text{if } \Delta\mathcal{H} \leq Th \\ 0 & \text{otherwise} \end{cases}$$

## Great Deluge Algorithm

$$W(\sigma \rightarrow \tau) = \begin{cases} 1 & \text{if } \mathcal{H}(\tau) \leq \mathcal{T} \\ 0 & \text{otherwise} \end{cases}$$

## Penna Criterion

$$W(\sigma \rightarrow \tau) = \min \left\{ 1, \left( 1 - (1 - q) \frac{\Delta\mathcal{H}}{kT} \right)^{\frac{1}{1-q}} \right\}$$

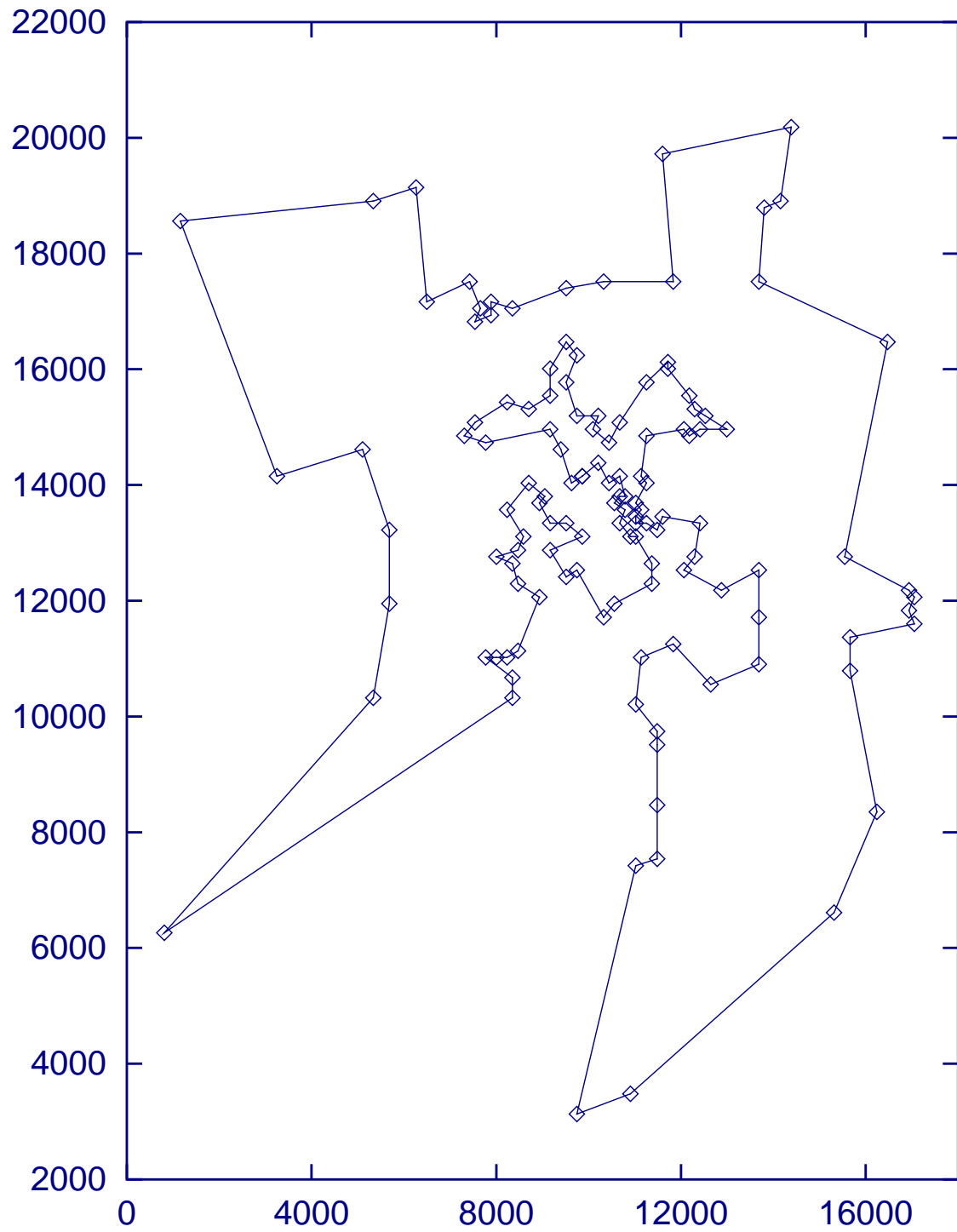
Working Horse

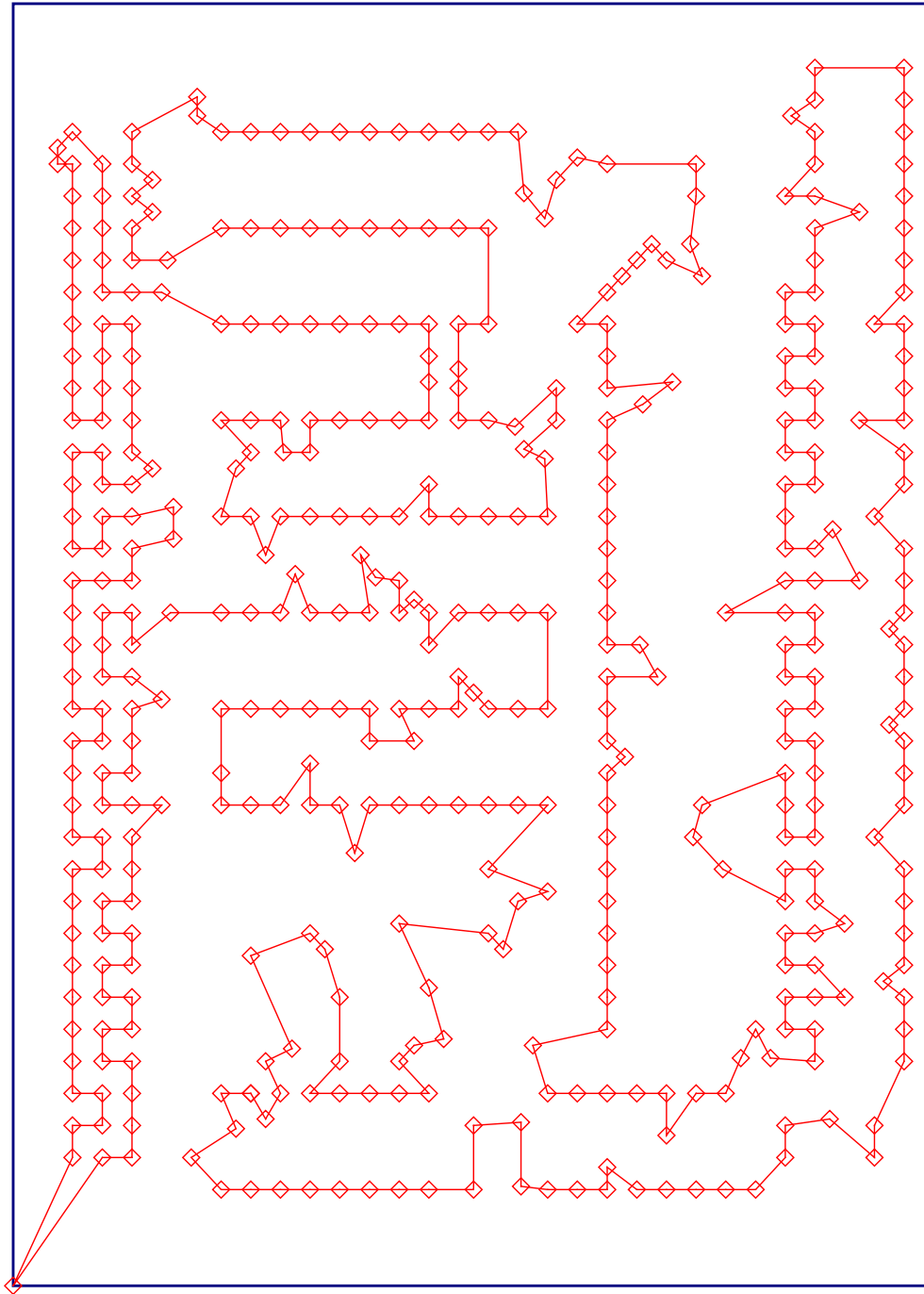
The Standard Problem of Optimization

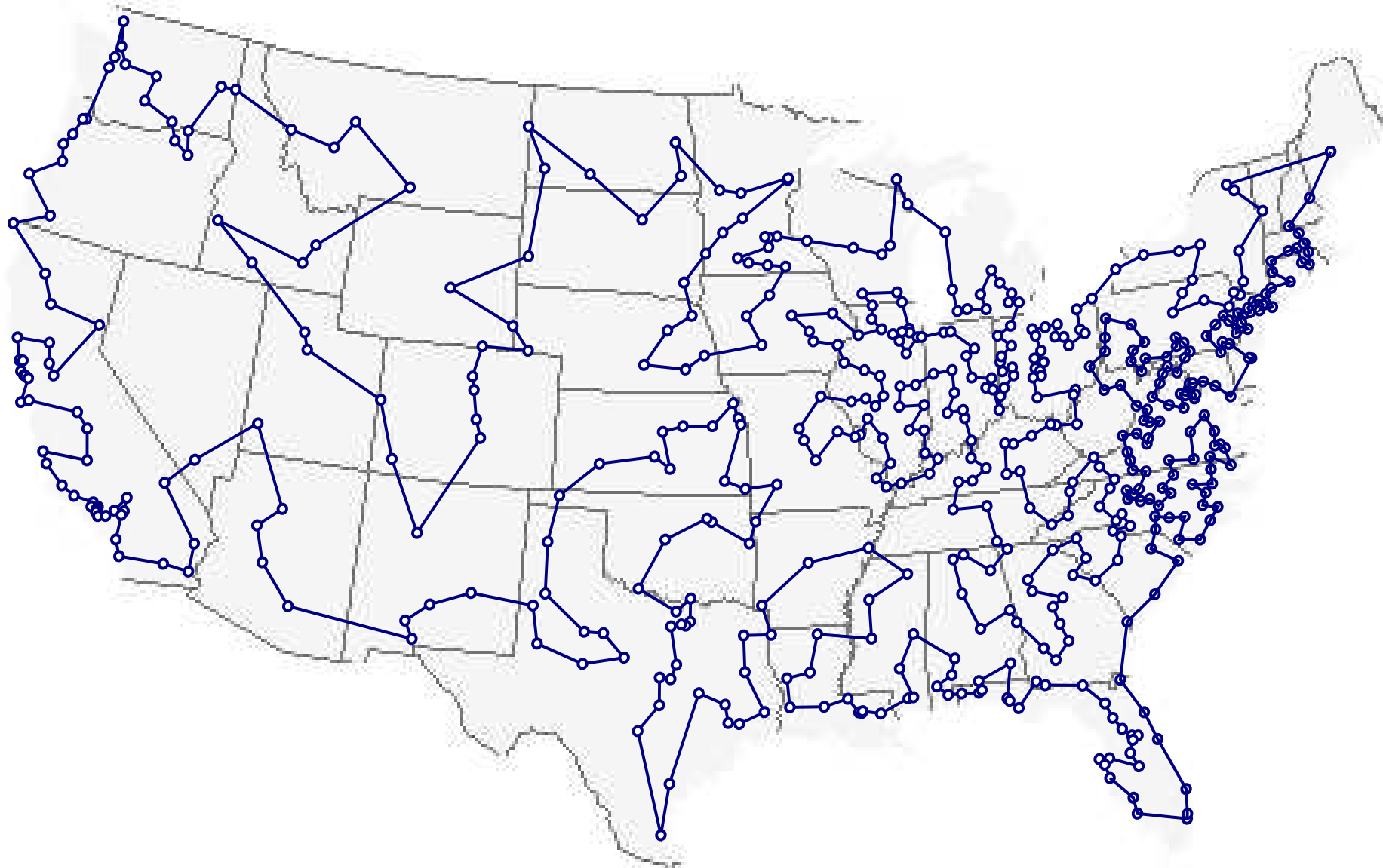
The Traveling Salesman Problem (TSP)

Benchmark libraries

Benchmark instances like BEER127, PCB442, and ATT532

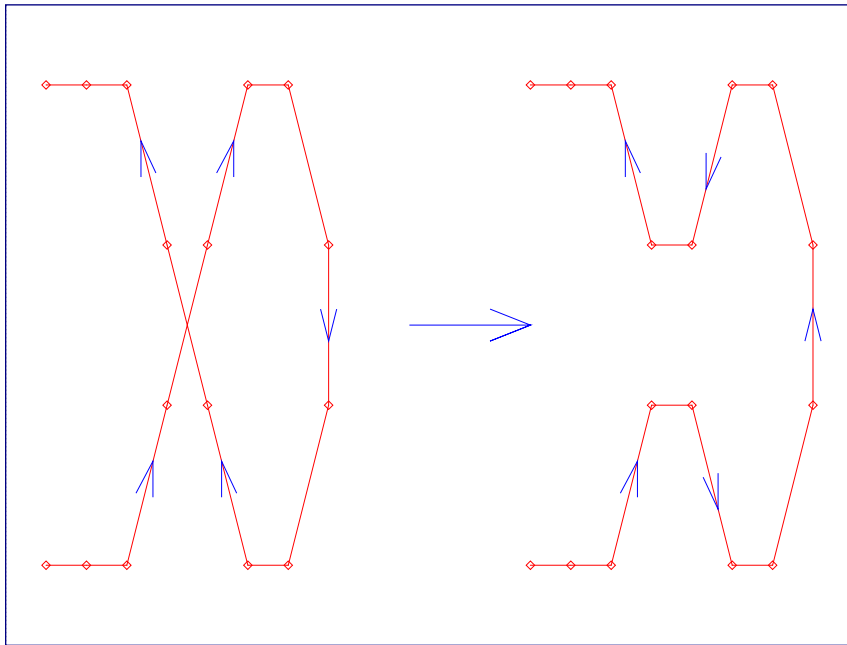




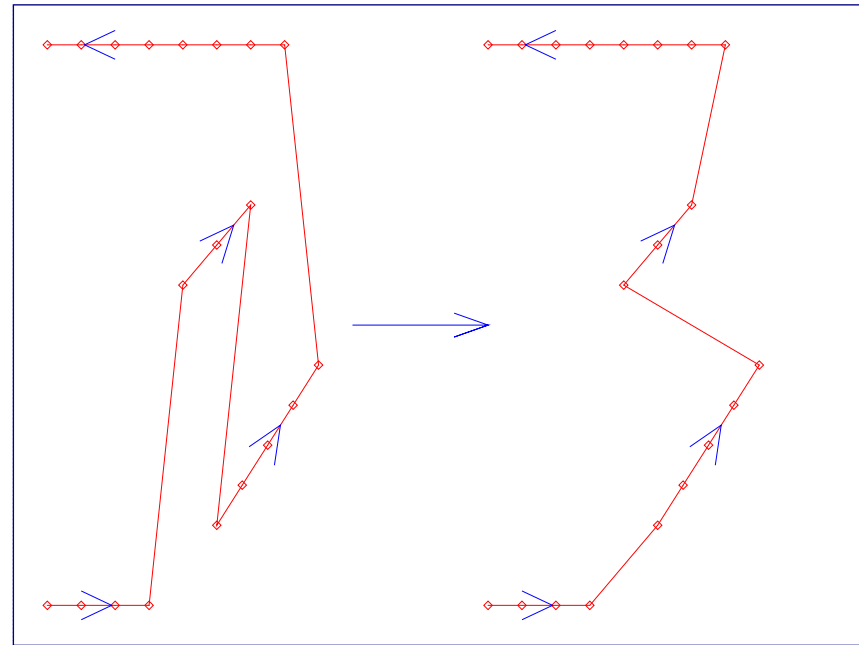


# Moves for the TSP

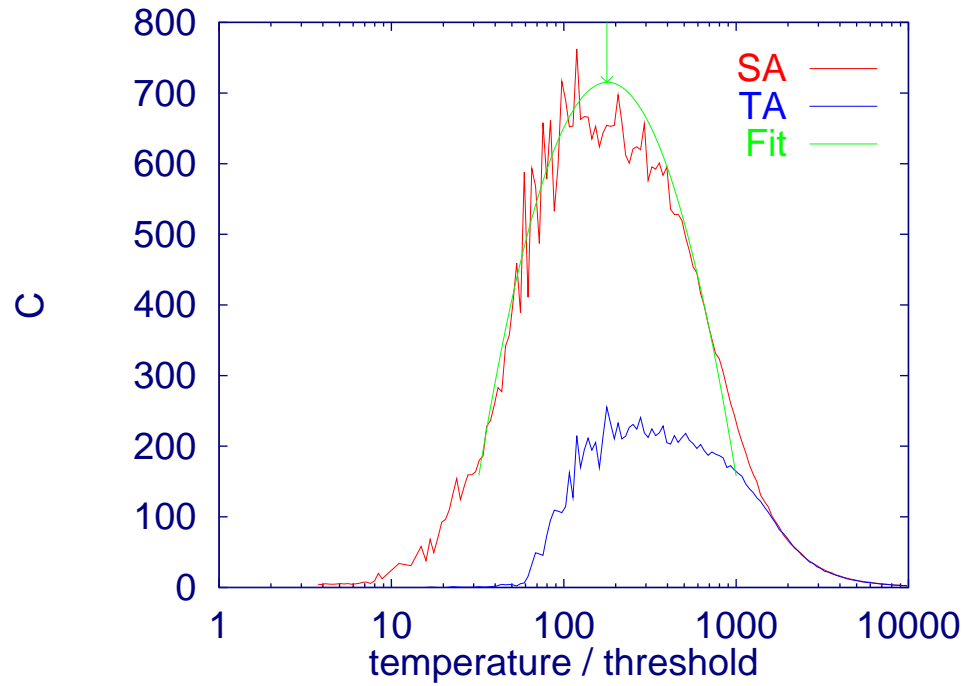
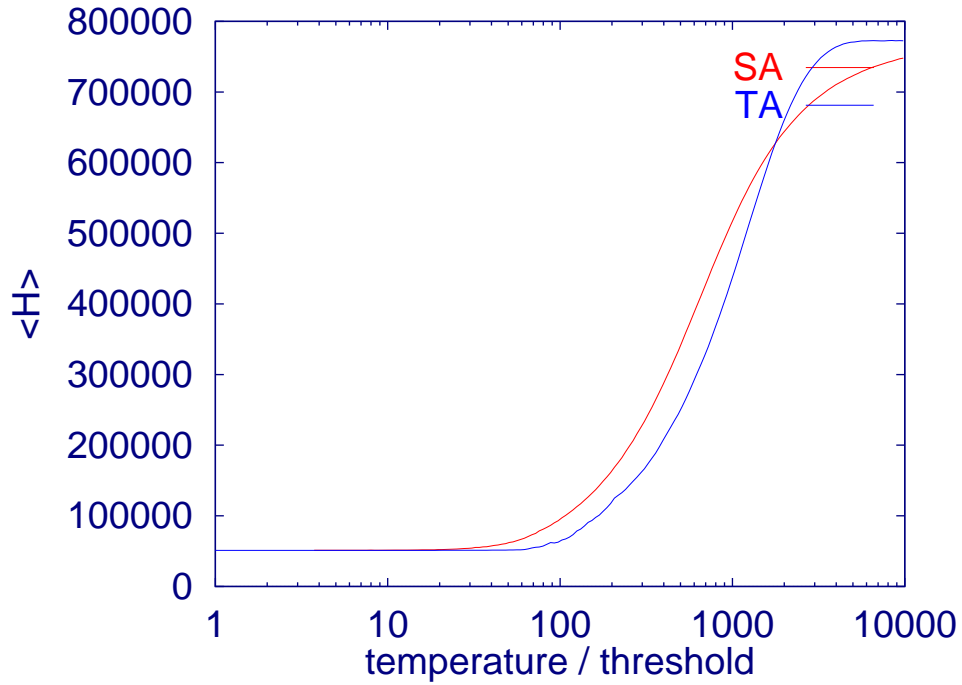
## Lin-2-Opt



## Lin-3-Opt



# Results



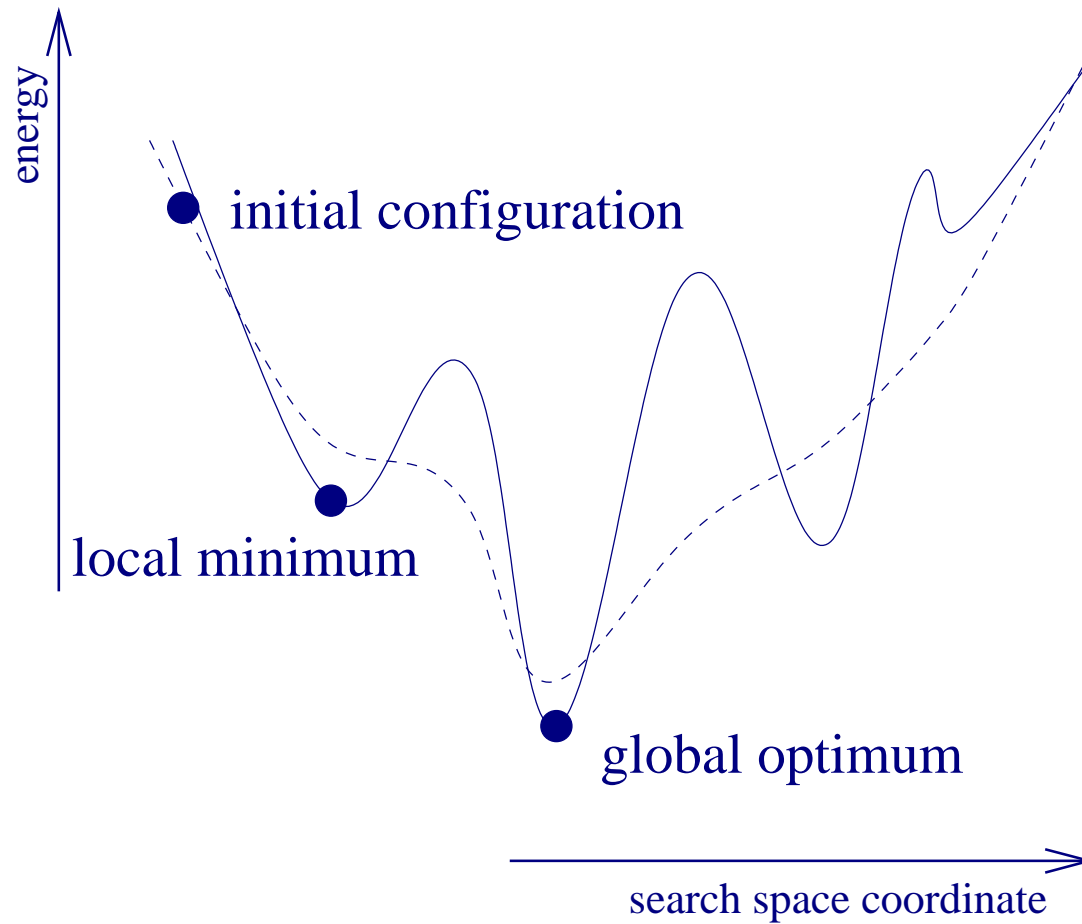
$$\langle \mathcal{H} \rangle = \frac{1}{Z} \sum_{\sigma} \mathcal{H}(\sigma) \exp \left( -\frac{\mathcal{H}(\sigma)}{k_B T} \right)$$

$$C = \frac{\partial \langle \mathcal{H} \rangle}{\partial T} = \frac{\text{Var}(\mathcal{H})}{k_B T^2}$$

# Methods to Increase the Efficiency

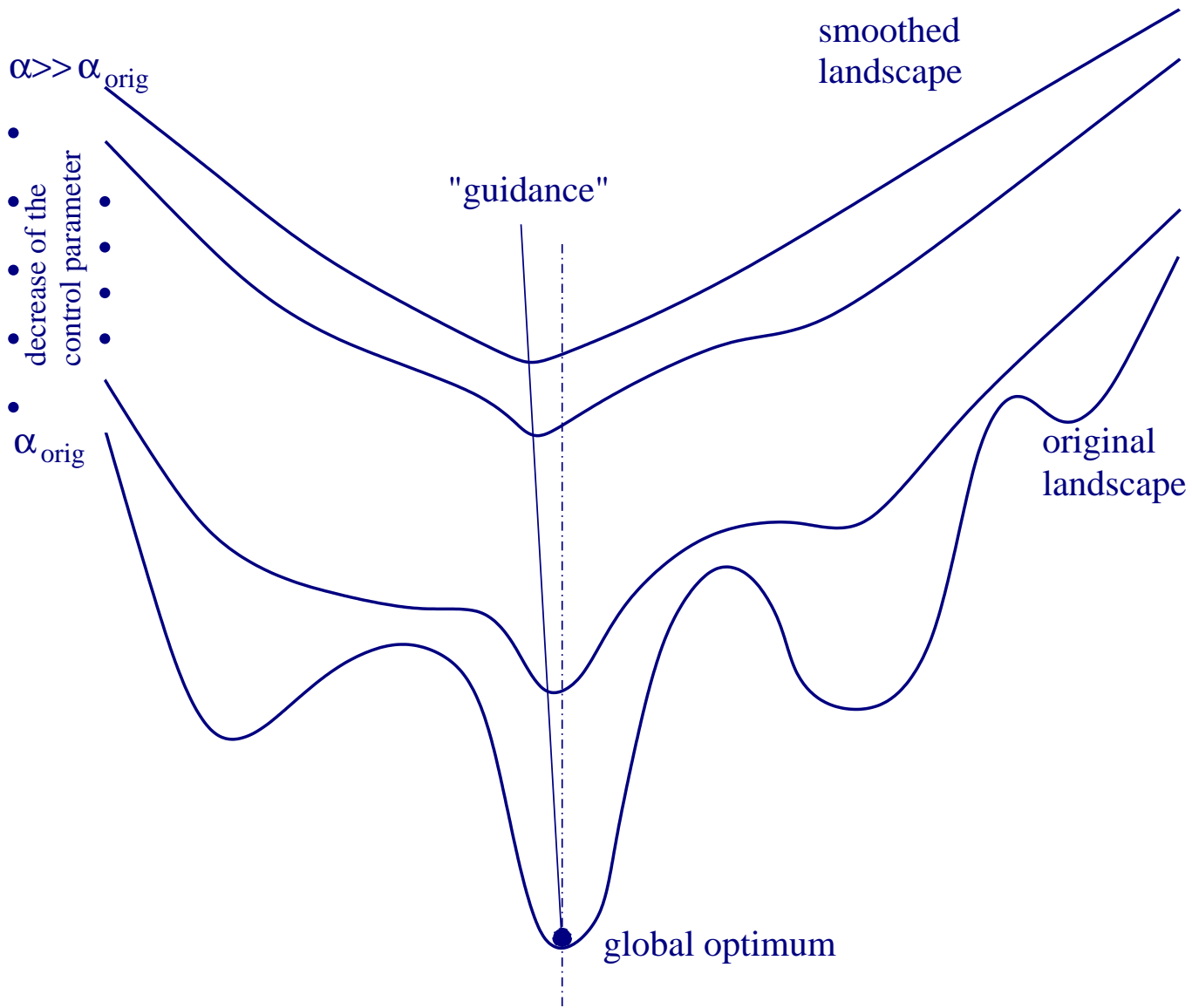
- Choice of the Control Parameter
- Choice of the Cooling Schedule
- Choice of the Moves

# Search Space Smoothing



— original landscape

- - - smoothed landscape



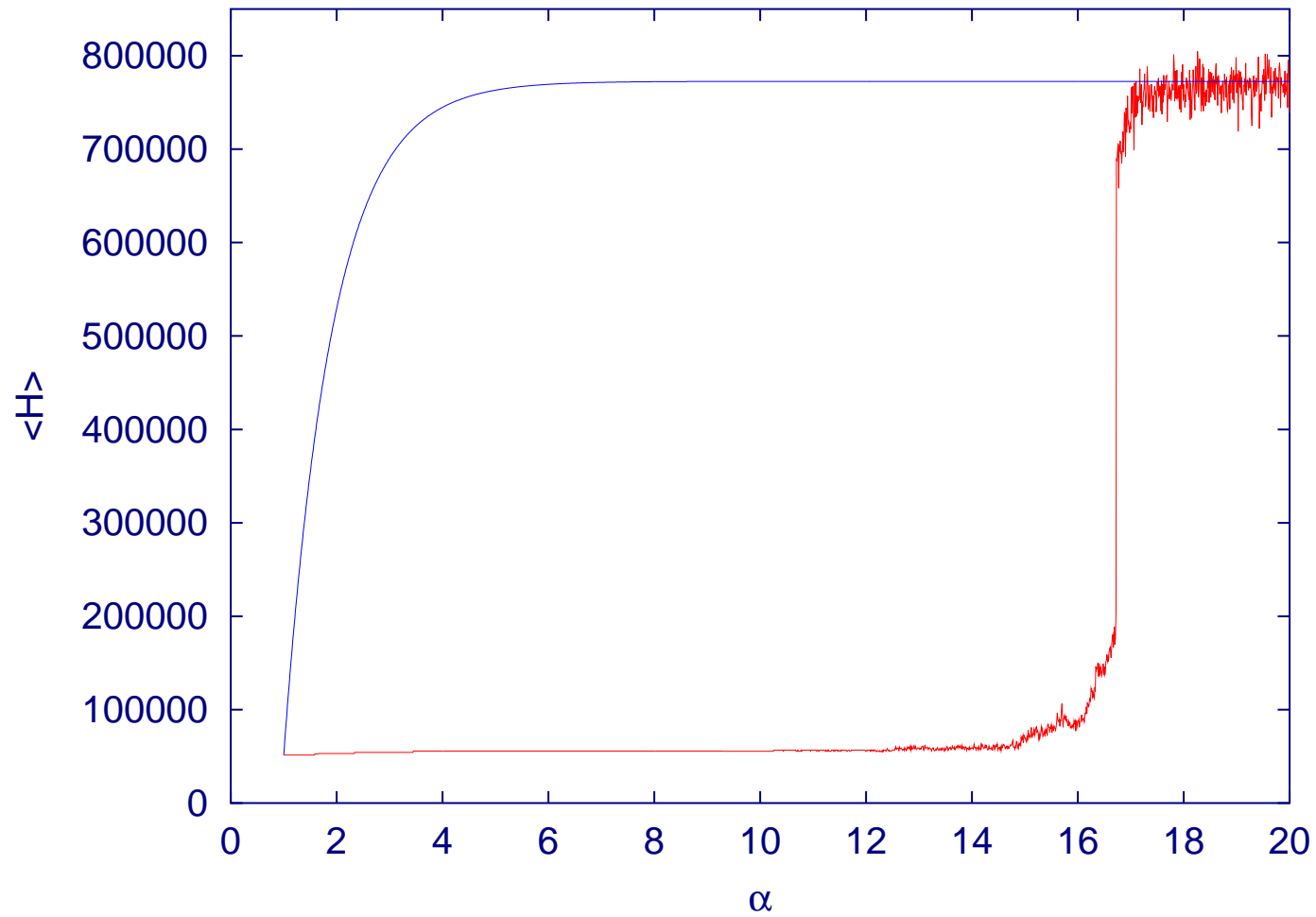
# Search Space Smoothing

normalized distances  $d(i, j) = D(i, j) / D_{\max}$

mean distance

$$\bar{d} = \frac{1}{N(N-1)} \sum_{i,j=1}^N d(i, j)$$

$$d_{\alpha}(i, j) = \begin{cases} \bar{d} + (d(i, j) - \bar{d})^{\alpha} & \text{if } d(i, j) \geq \bar{d} \\ \bar{d} - (\bar{d} - d(i, j))^{\alpha} & \text{otherwise} \end{cases}$$



red curve according to the original distances

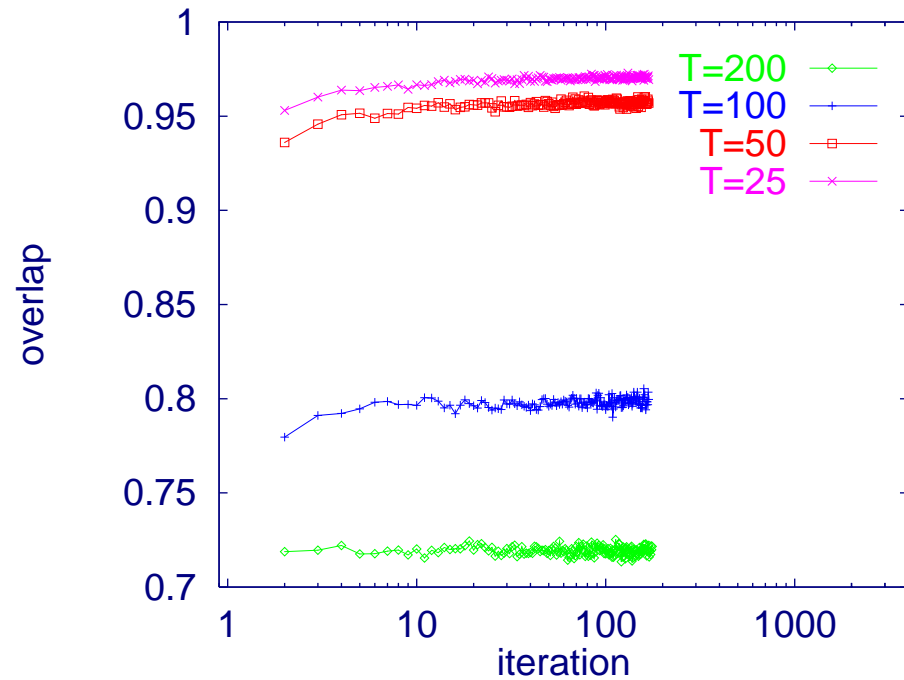
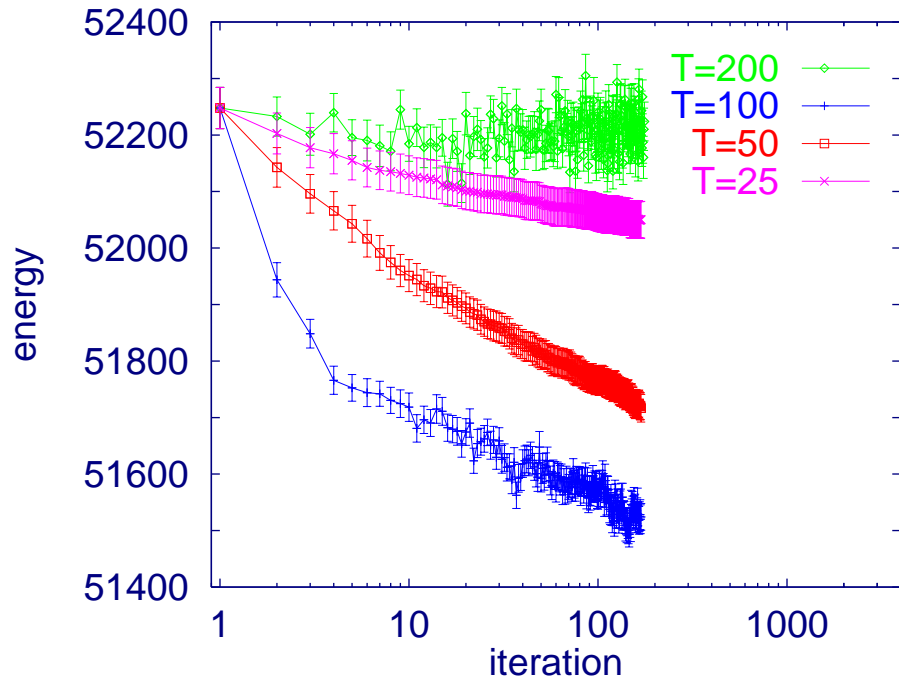
blue curve according to the unsmoothed distances

(unnormalized)

# Bouncing

classic ansatz:            monotonous cooling

new ansatz:                iterated cooling and  
                                 reheating of the system

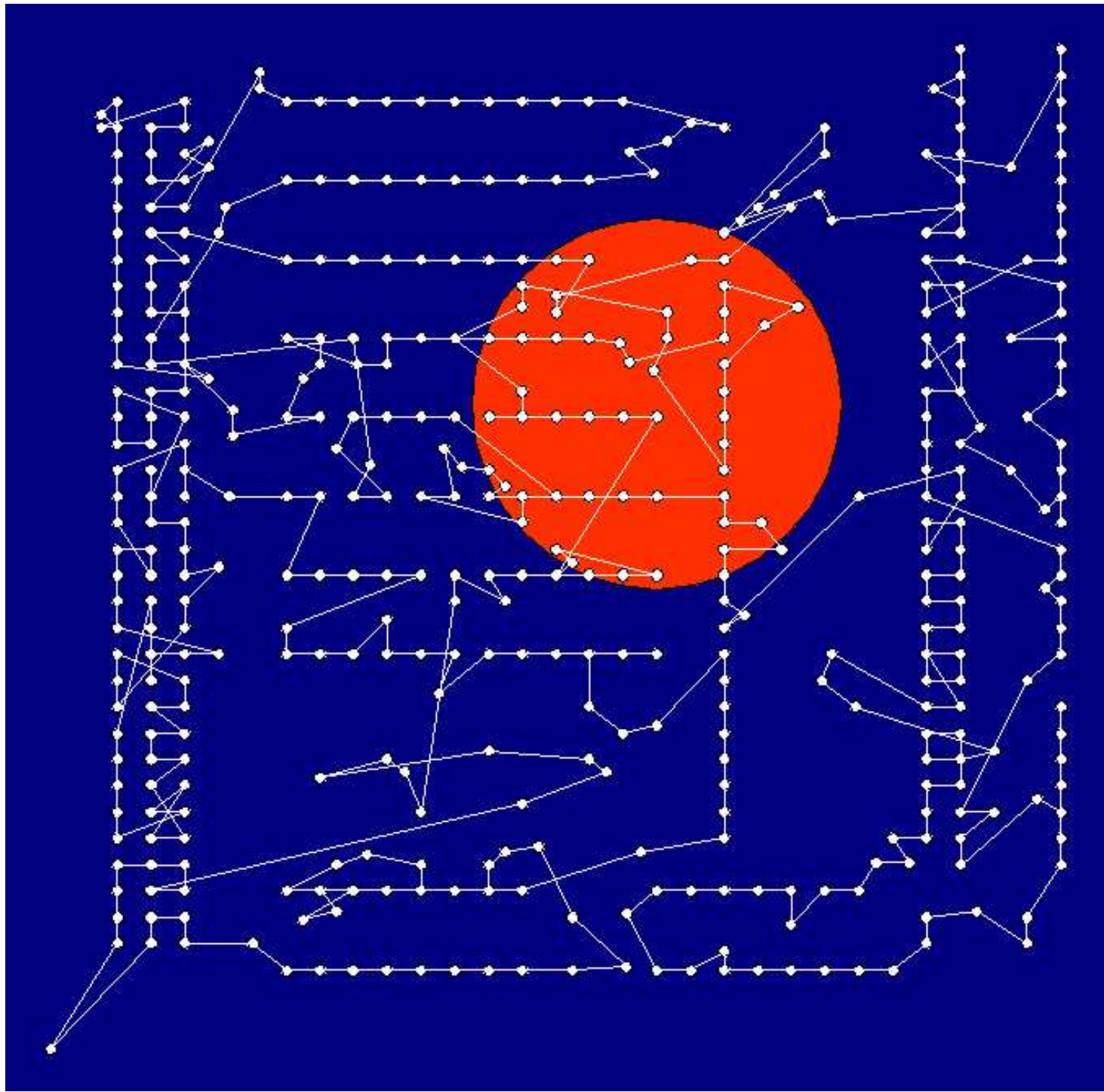


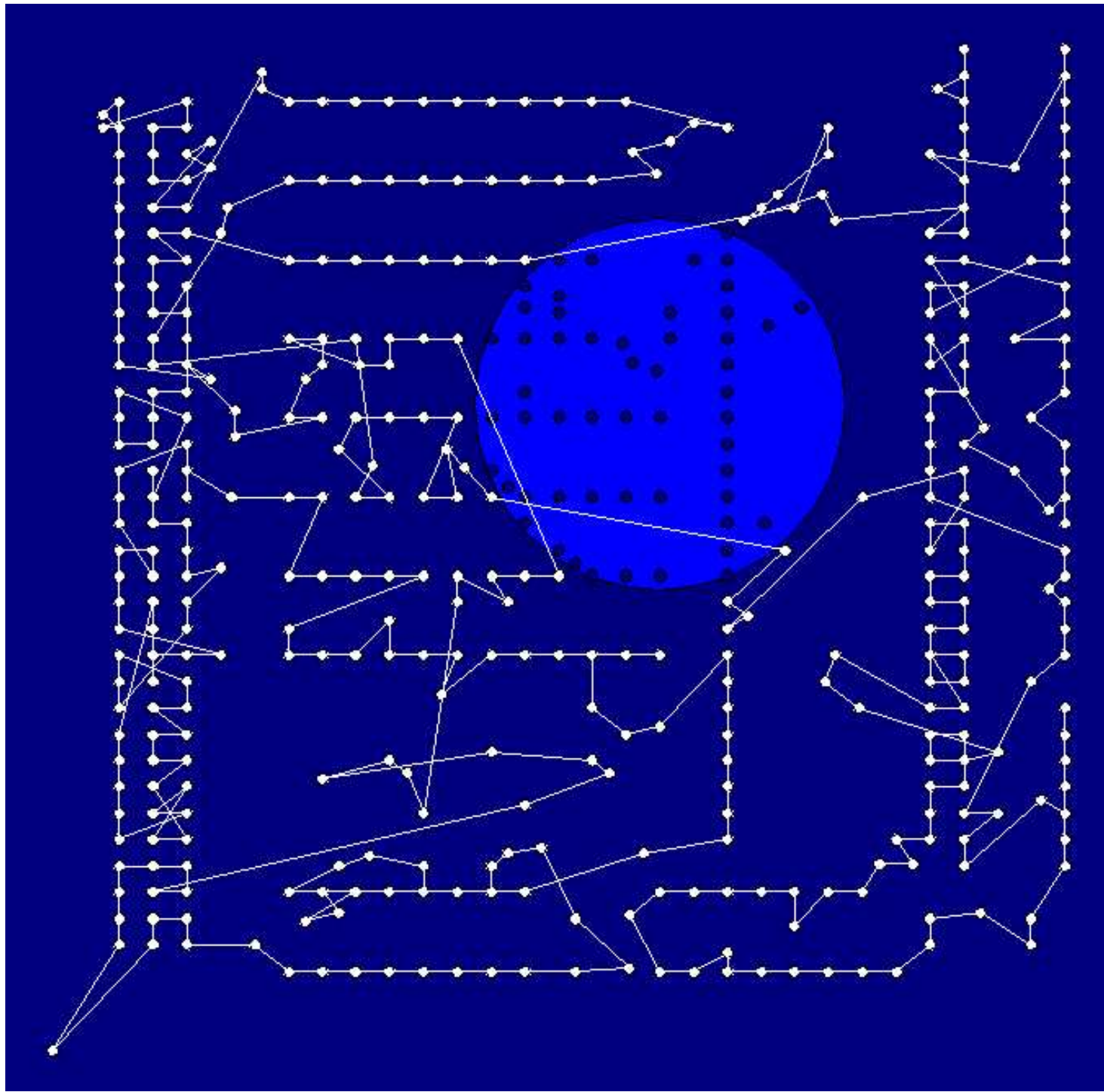


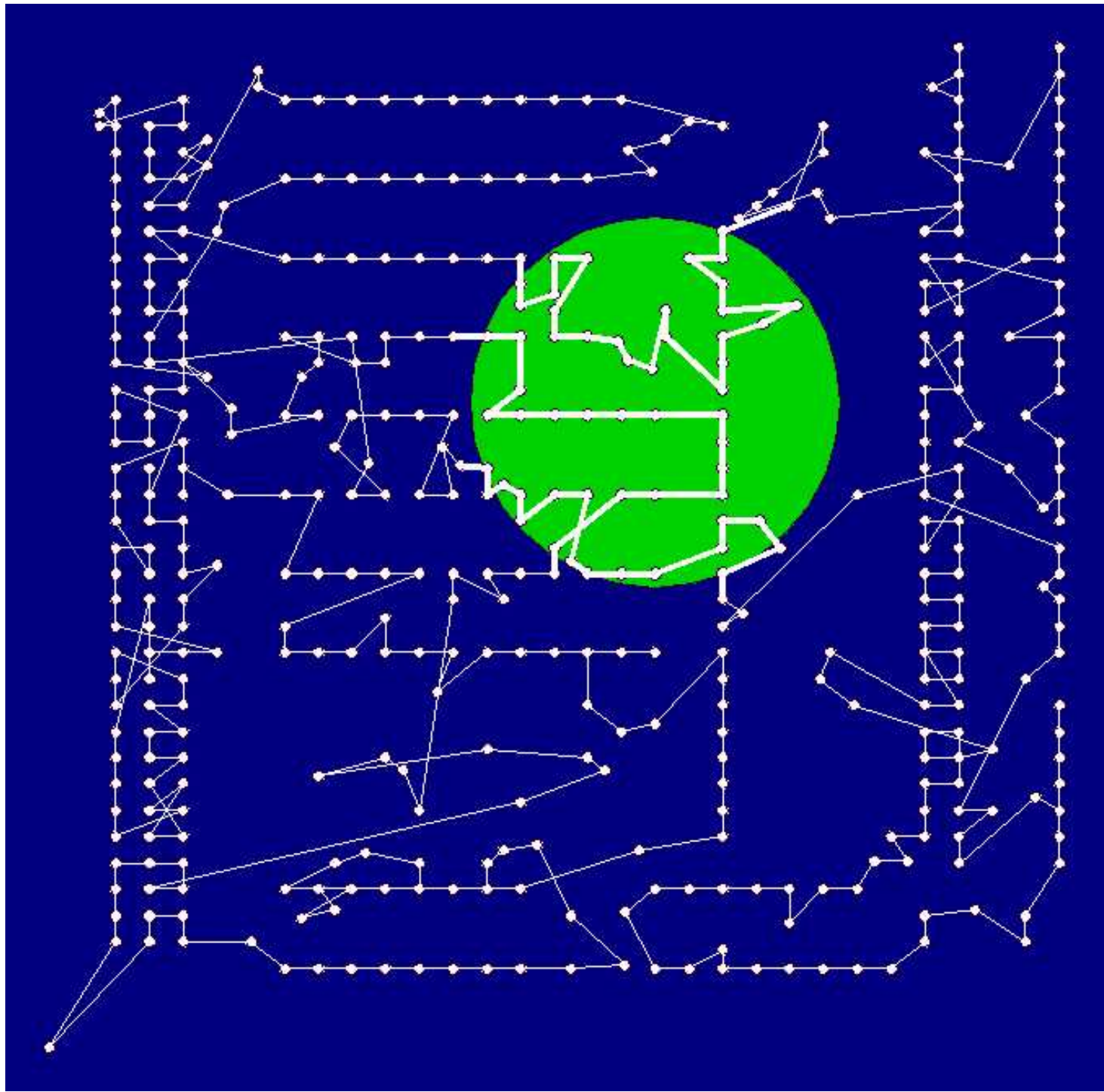
# Ruin & Recreate

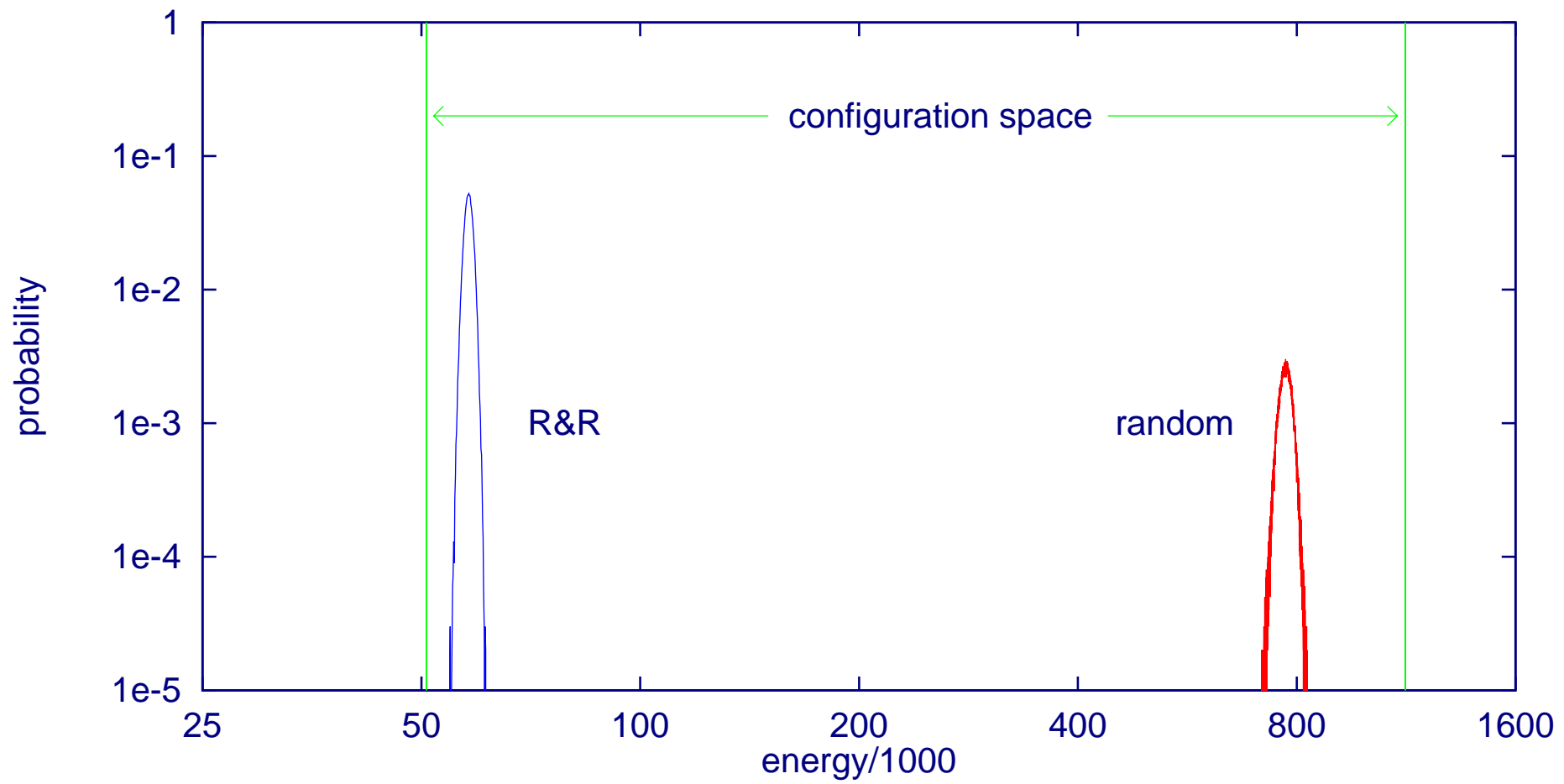
classic ansatz: small local change

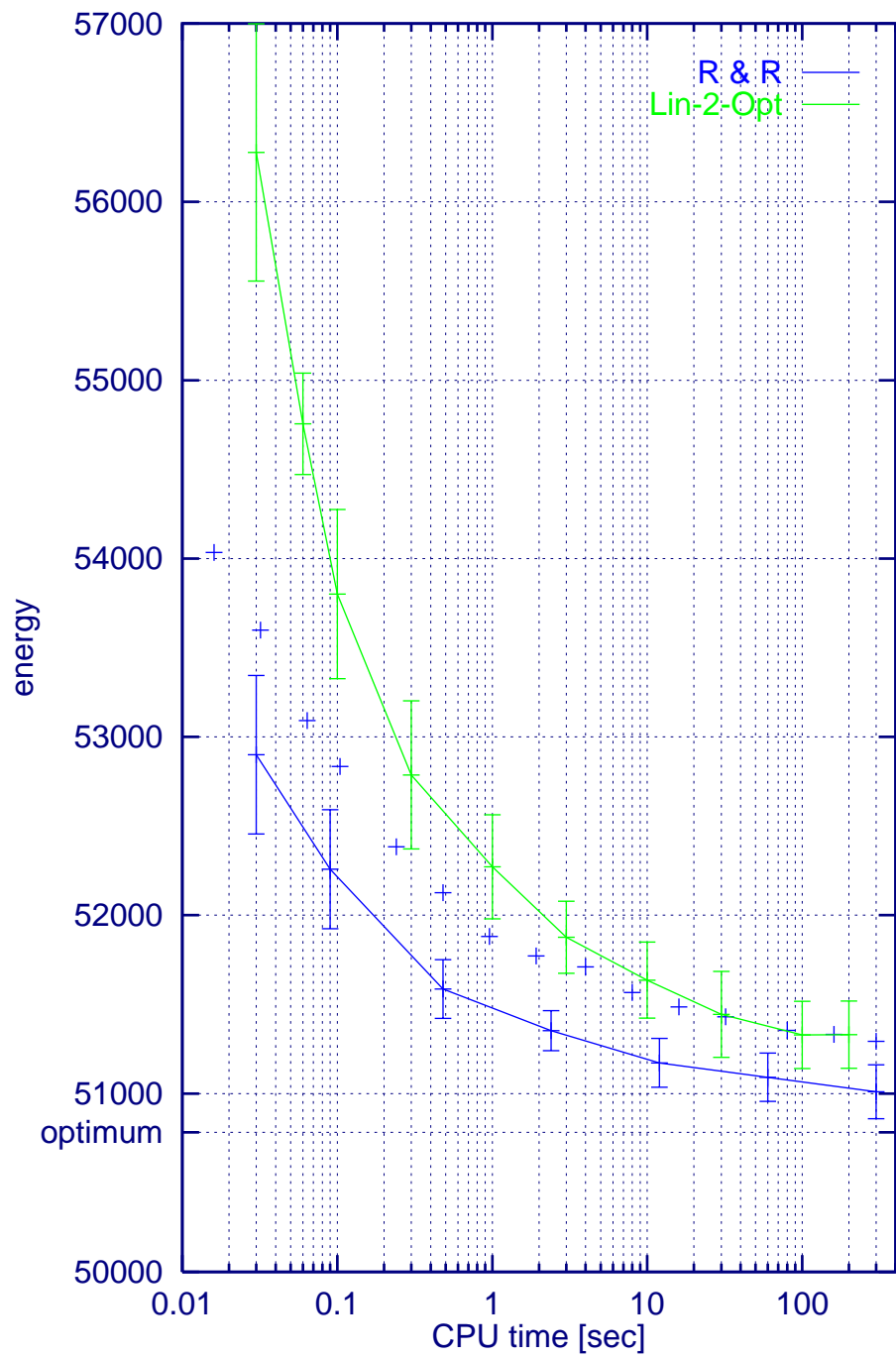
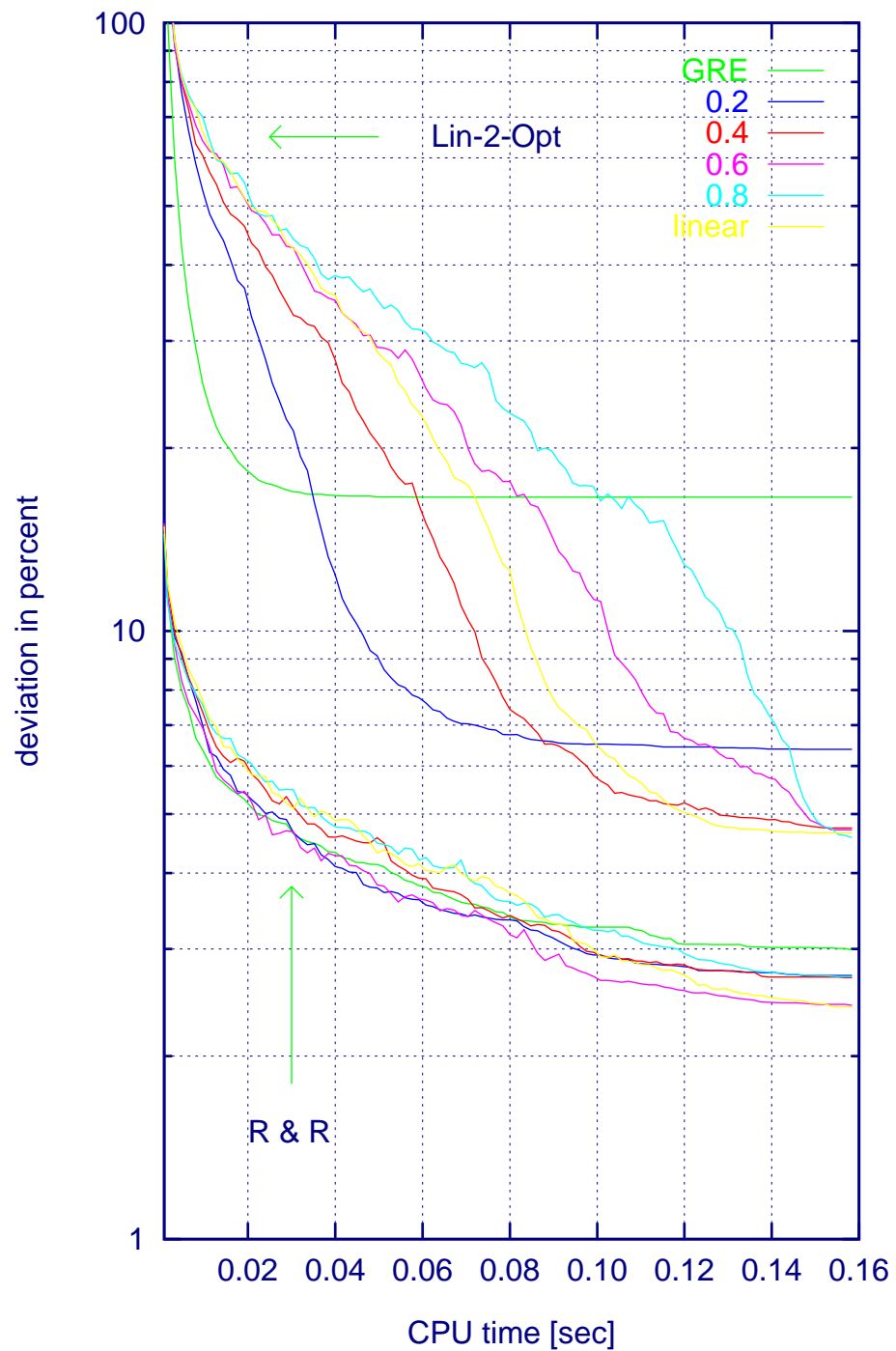
new ansatz: destruction  
and intelligent recreation  
of a larger system part







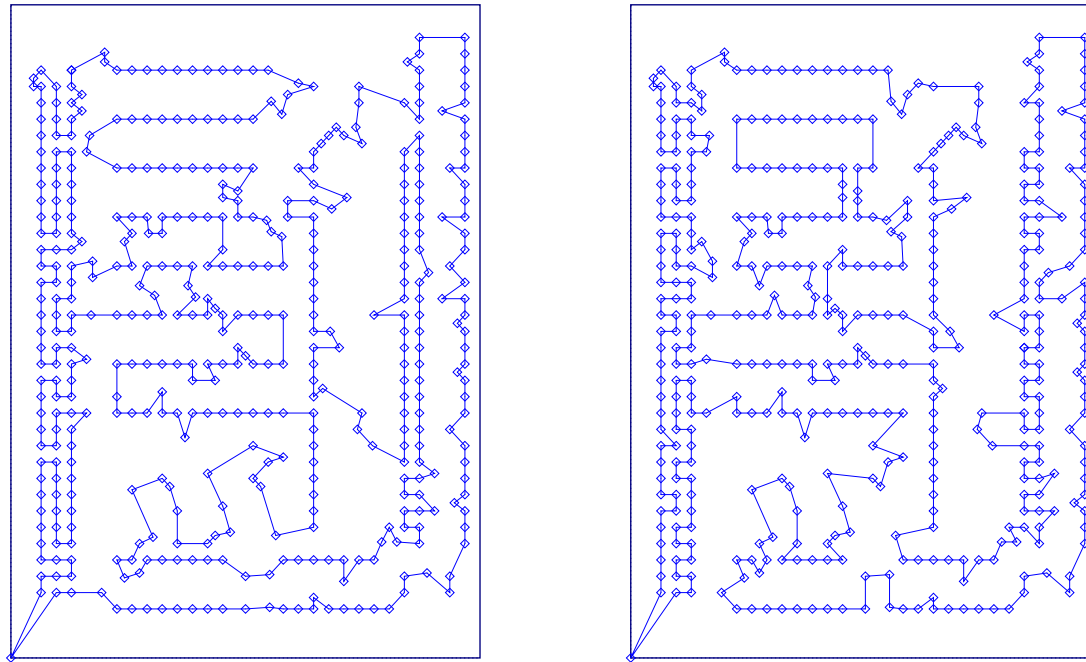




# Approaches for Parallelization

- Splitting of the Problem into Subproblems
- Information Exchange during the Optimization Run
- Information Exchange after the Optimization Run

# Searching for Backbones

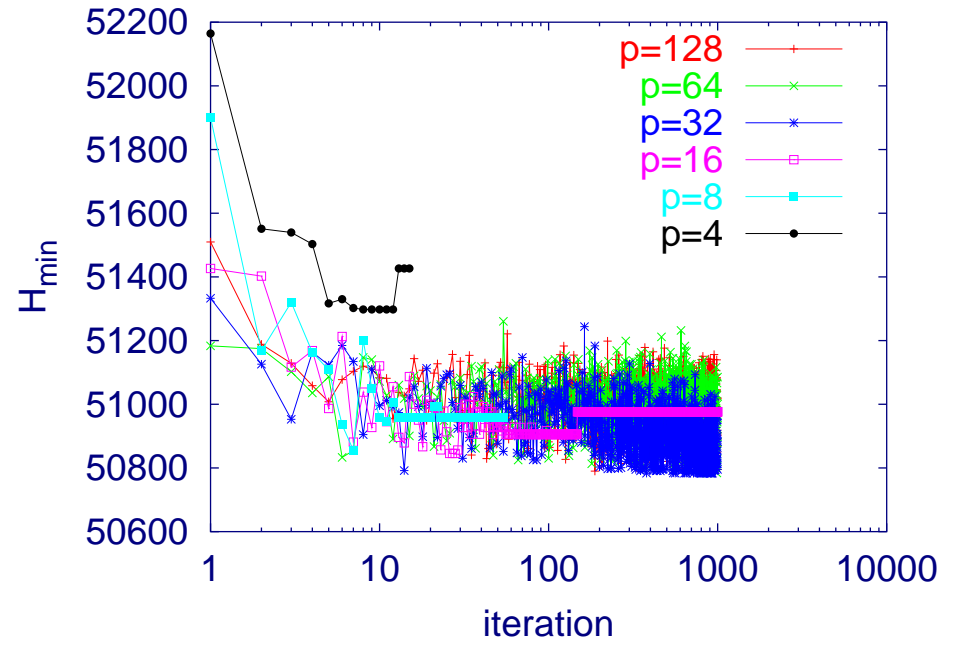
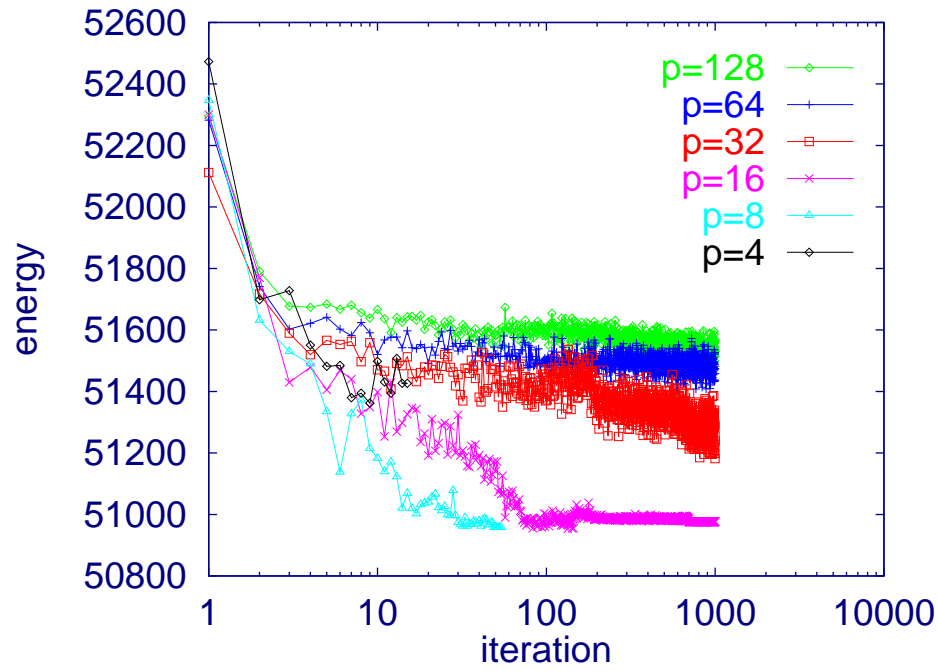


Finding common structures in different solutions and eliminating them in order to save calculation time and achieve better solutions

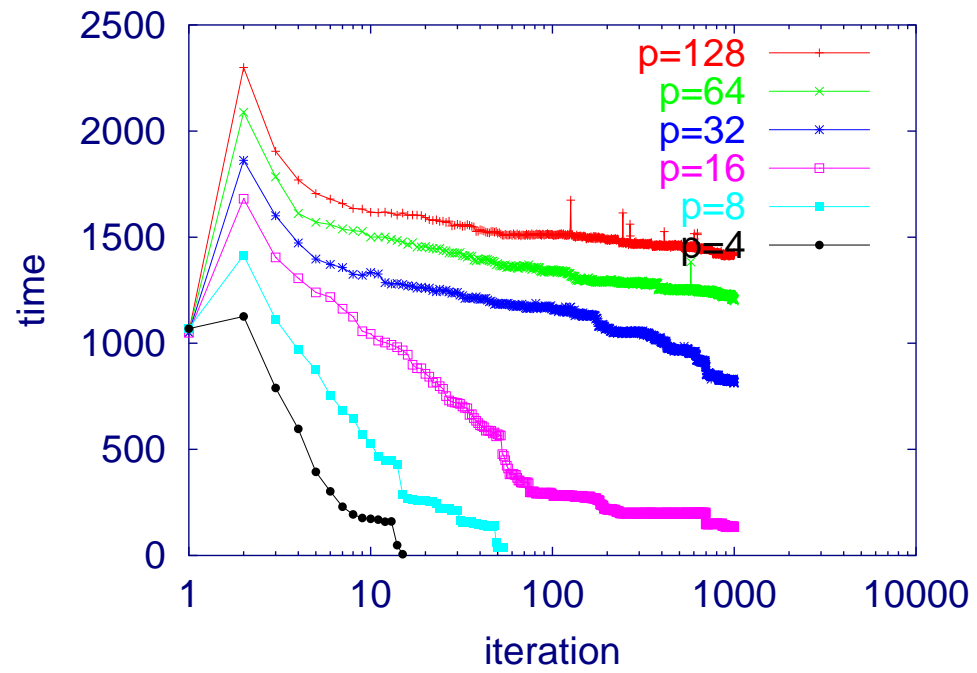
# Outline of the Searching for Backbones Algorithm

1. Perform several independent optimization runs.
2. Compare the achieved solutions for common parts.
3. Assume these common parts to be optimally solved and to be part of the optimum solution.
4. Perform again several independent optimization runs, but now keeping the found backbones constant during the optimization run.
5. If the solutions differ, return to step 2.

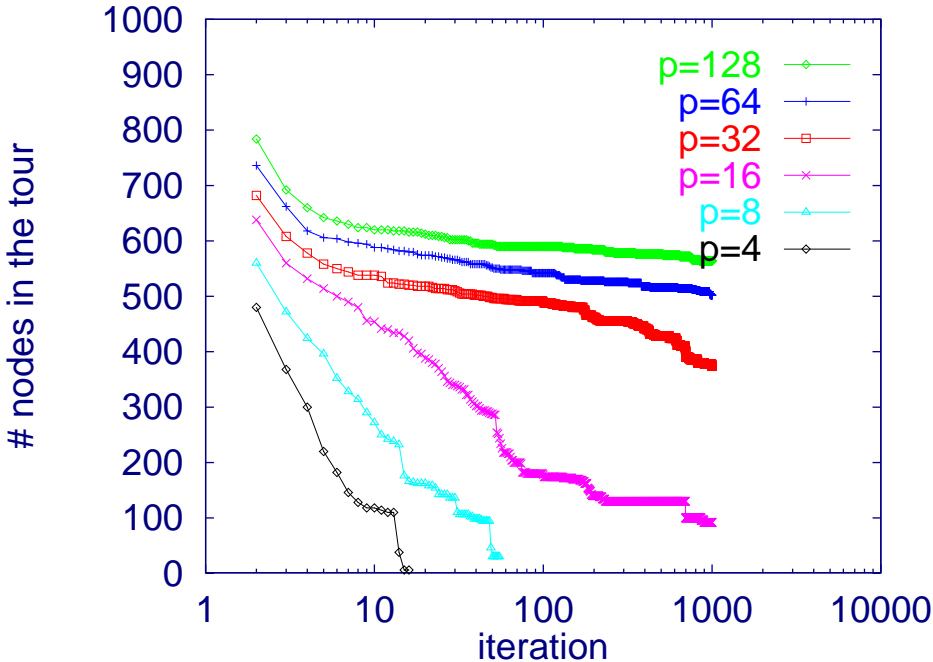
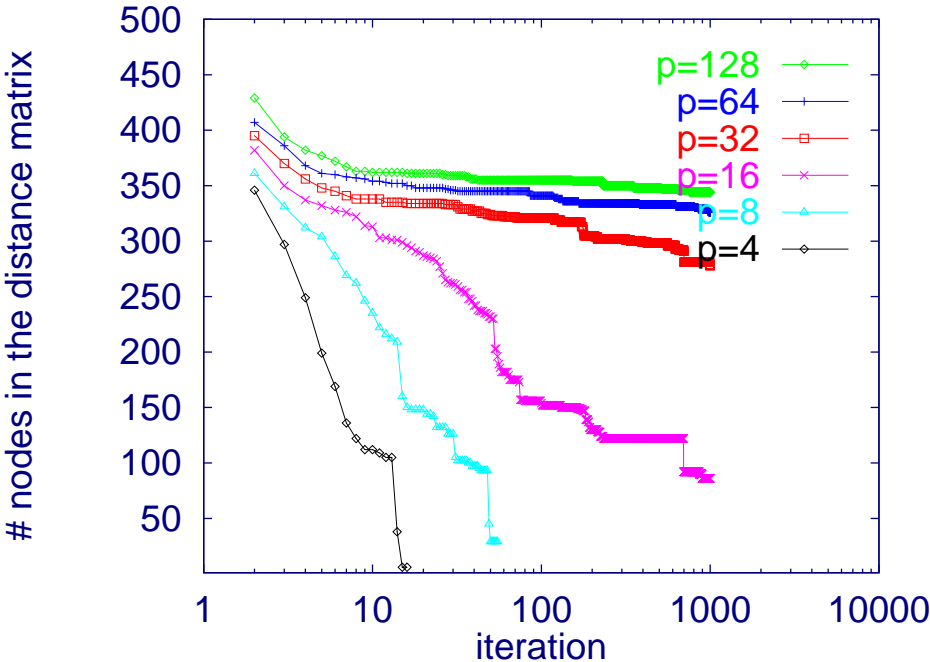
# Quality of the Results



# Calculation Time



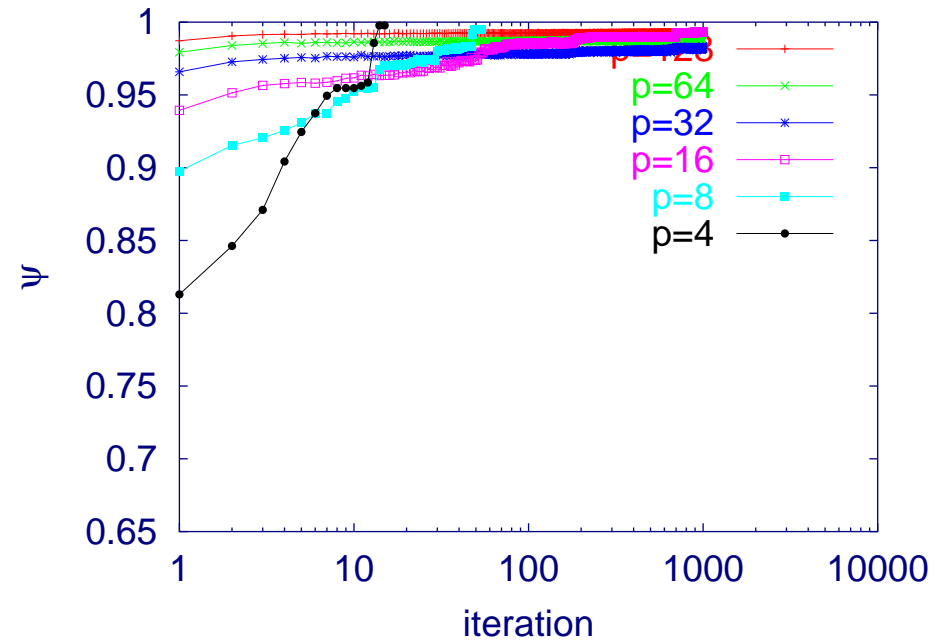
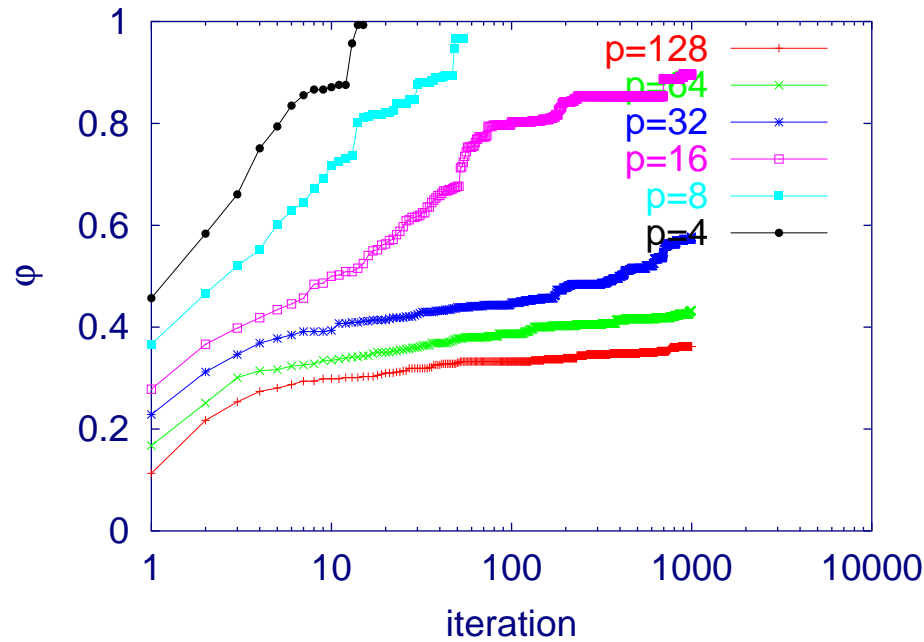
# Convergence behavior of the algorithm



# Order Parameters

$$\varphi(\eta) = \frac{\sum_{i,j} \delta_{\eta(i,j),p}}{2N}$$

$$\psi(\eta) = 1 - \frac{-2N + \sum_{i,j} (1 - \delta_{\eta(i,j),0})}{2N(p-1)}$$

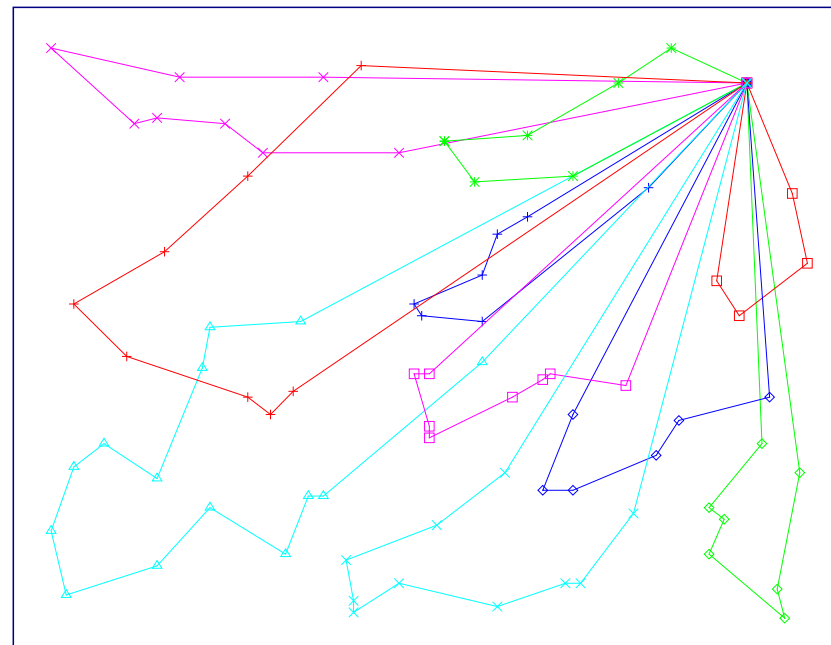


# Applications to Other Problems

## Vehicle Routing Problems

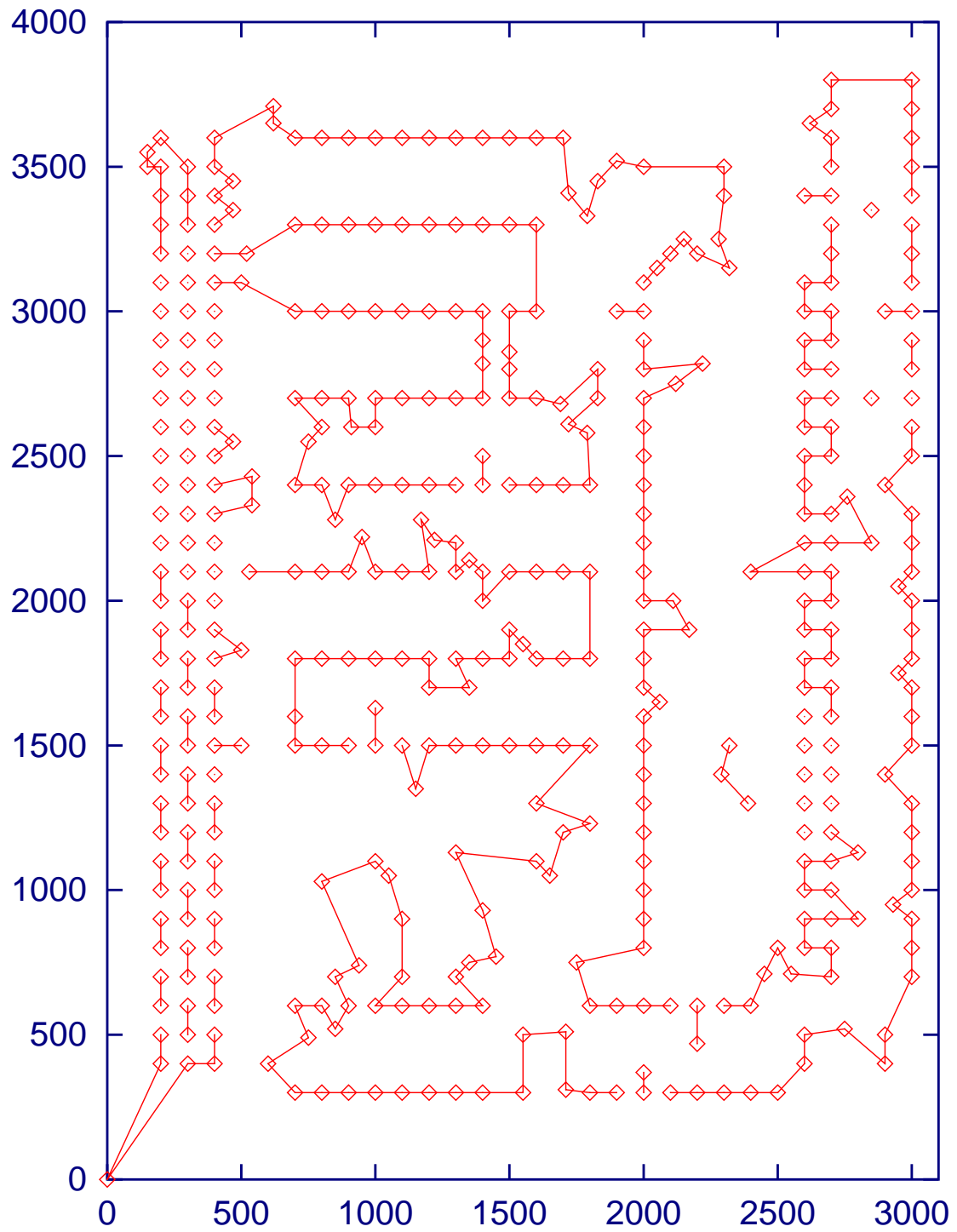
TSP → MTSP → VRP

A-n80-k10



Add a **penalty function** with a Lagrange multiplier  $\lambda$  to fulfill the capacity constraint:

$$\begin{aligned}\mathcal{H}(\sigma) &= \sum_{l=1}^L \sum_{i=1}^{N_l-1} D(\sigma(i, l), \sigma(i+1, l)) \\ &+ \lambda \sum_{l=1}^L \left( \sum_{i=2}^{N_l-1} m(\sigma(i, l)) - \kappa + \gamma \right) \\ &\ominus \left( \sum_{i=2}^{N_l-1} m(\sigma(i, l)) - \kappa \right)\end{aligned}$$

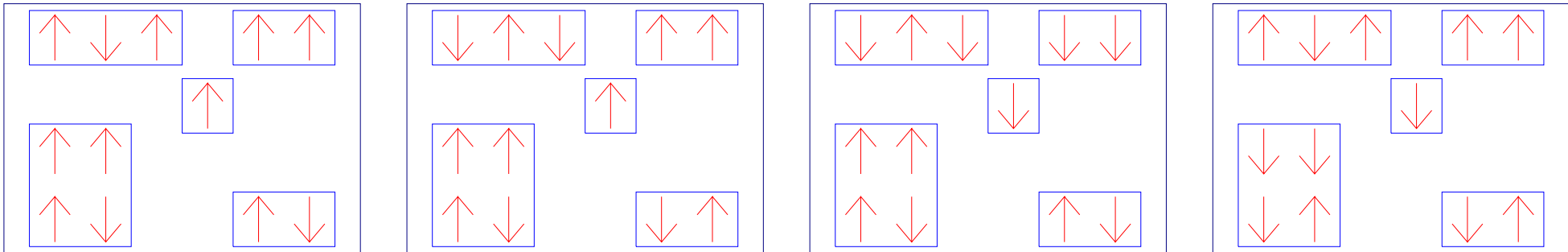


# Spin Glasses

## SK model

$$\mathcal{H} = - \sum_{i,j} J_{ij} S_i S_j$$

with  $S_i = \pm 1$  and  $P(J_{ij}) \propto \exp(-J_{ij}^2/2)$



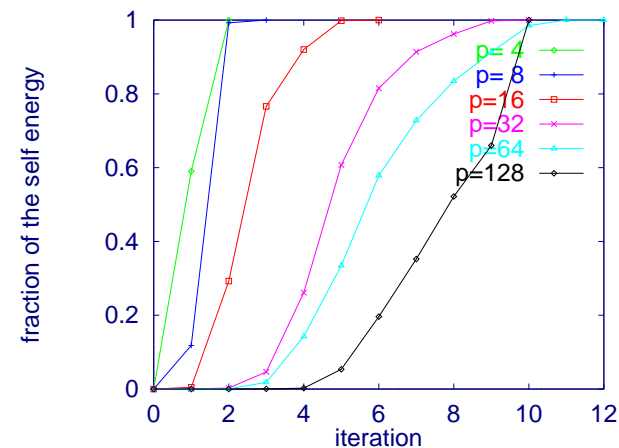
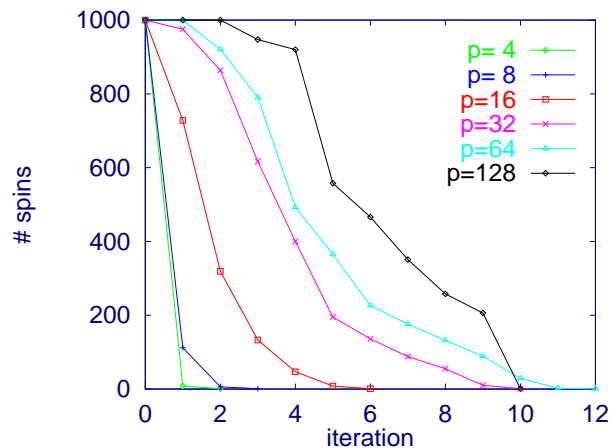
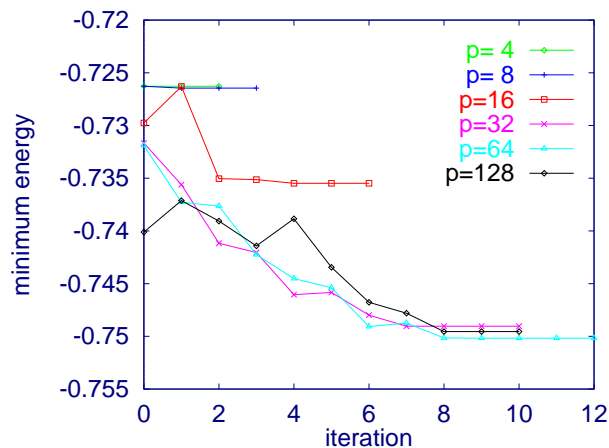
# Searching for Backbones

$$\eta(i, j) = \left| \sum_{\nu=1}^p S_i^{\nu} \cdot S_j^{\nu} \right|$$

Spins  $i$  and  $j$  can be put together to one backbone spin  $\alpha$  if  $\eta(i, j) = p$ .

Hamiltonian for backbone spins:

$$\mathcal{H} = - \sum_{\alpha, \beta} J_{\alpha\beta} S_{\alpha} S_{\beta} \quad \text{with} \quad J_{\alpha\beta} S_{\alpha} S_{\beta} = \sum_{\substack{i \in \alpha \\ j \in \beta}} J_{ij} S_i^{\xi} S_j^{\xi}$$



# References

- (1) J. Schneider, Ch. Froschhammer, I. Morgenstern, Th. Husslein, J. M. Singer, Searching for Backbones — An Efficient Parallel Algorithm for the Traveling Salesman Problem, *Comp. Phys. Comm.* **96**, 173-188, 1996.
- (2) J. Schneider, M. Dankesreiter, W. Fettes, I. Morgenstern, M. Schmid, J. M. Singer, Search Space Smoothing for Combinatorial Optimization Problems, *Phys. A* **243**, 77-112, 1997.
- (3) J. Schneider, I. Morgenstern, J. M. Singer, Bouncing towards the optimum: Improving the results of Monte Carlo optimization algorithms, *Phys. Rev. E* **58**, 5085-5095, 1998.
- (4) G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, G. Dueck, Record Breaking Optimization Results — Using the Ruin &

Recreate Principle, J. Comp. Phys. **159**, 139-171, 2000.

- (5) J. Schneider, J. Britze, A. Ebersbach, I. Morgenstern, M. Puchta, Optimization of Production Planning Problems — A Case Study for Assembly Lines, Int. J. Mod. Phys. C **11**, 949-972, 2000.
- (6) J. Schneider, Searching for Backbones – a high-performance parallel algorithm for solving combinatorial optimization problems, Future Generation Computer Systems **19**, 121-131, 2003.