Ant Colony Optimization

Part 3: Algorithms

Spring 2009

Instructor: Dr. Masoud Yaghini

Outline

- The Traveling Salesman Problem
- ACO Algorithms for TSP
- Ant System
- Elitist Ant System
- Rank-based Ant System
- MAX–MIN Ant System
- Ant Colony System
- Search Stagnation
- Experimental Evaluation
- ACO plus Local Search
- References

The Traveling Salesman Problem

The Traveling Salesman Problem

- The traveling salesman problem is an **extensively studied** problem in the literature.
- The TSP also plays an important role in ACO research: the first ACO algorithm, called **Ant System**, as well as many of the ACO algorithms proposed subsequently, was first tested on the TSP.

The Traveling Salesman Problem

- The reasons for the choice of the TSP:
  - it is an important NP-hard optimization problem that arises in several applications
  - it is a problem to which ACO algorithms are easily applied
  - it is easily understandable, so that the algorithm behavior is not obscured by too many technicalities
  - it is a standard test bed for new algorithmic ideas—a good performance on the TSP is often taken as a proof of their usefulness
  - the most efficient ACO algorithms for the TSP were also found to be among the most efficient ones for a wide variety of other problems

The Traveling Salesman Problem

- The traveling salesman problem is the problem faced by:
  - a salesman who, starting from his home town,
  - wants to find a shortest possible trip through a given set of customer cities,
  - visiting each city once
  - finally returning home.

The Traveling Salesman Problem

- The TSP can be represented by a complete weighted graph G =(N, A) with:
  - N : the set of n = |N| nodes (cities)
  - A : the set of arcs fully connecting the nodes.
- Each arc is assigned a weight d_{ij} which represents the distance between cities i and j.
- The TSP is the problem of finding a minimum length **Hamiltonian circuit** of the graph, where a Hamiltonian circuit is a closed walk (a tour) visiting each node of G exactly once.

The Traveling Salesman Problem

- We may distinguish between:
  - **Symmetric** TSPs, where the distances between the cities are independent of the direction of traversing the arcs, that is, $d_{ij} = d_{ji}$ for every pair of nodes, and
  - **Asymmetric** TSP (ATSP), where at least for one pair of nodes i, j we have $d_{ij} \neq d_{ji}$

The Traveling Salesman Problem

- A solution to an instance of the TSP can be represented as a permutation of the city indices
- This permutation is cyclic, that is, the absolute position of a city in a tour is not important at all but only the relative order is important

Traveling Salesman Problem

- The only **constraint** in the TSP is that all cities have to be visited and that each city is visited at most once.
- This constraint is enforced if an ant at each construction step chooses the next city only among those it has not visited yet
- The feasible neighborhood of an ant k in city i, where k is the ant's identifier, comprises all cities that are still unvisited.

Traveling Salesman Problem

- Thus, an optimal solution to the TSP is a permutation π of the node indices {1, 2, . . ., n} such that the length f(π) is minimal, where f(π) is given by:

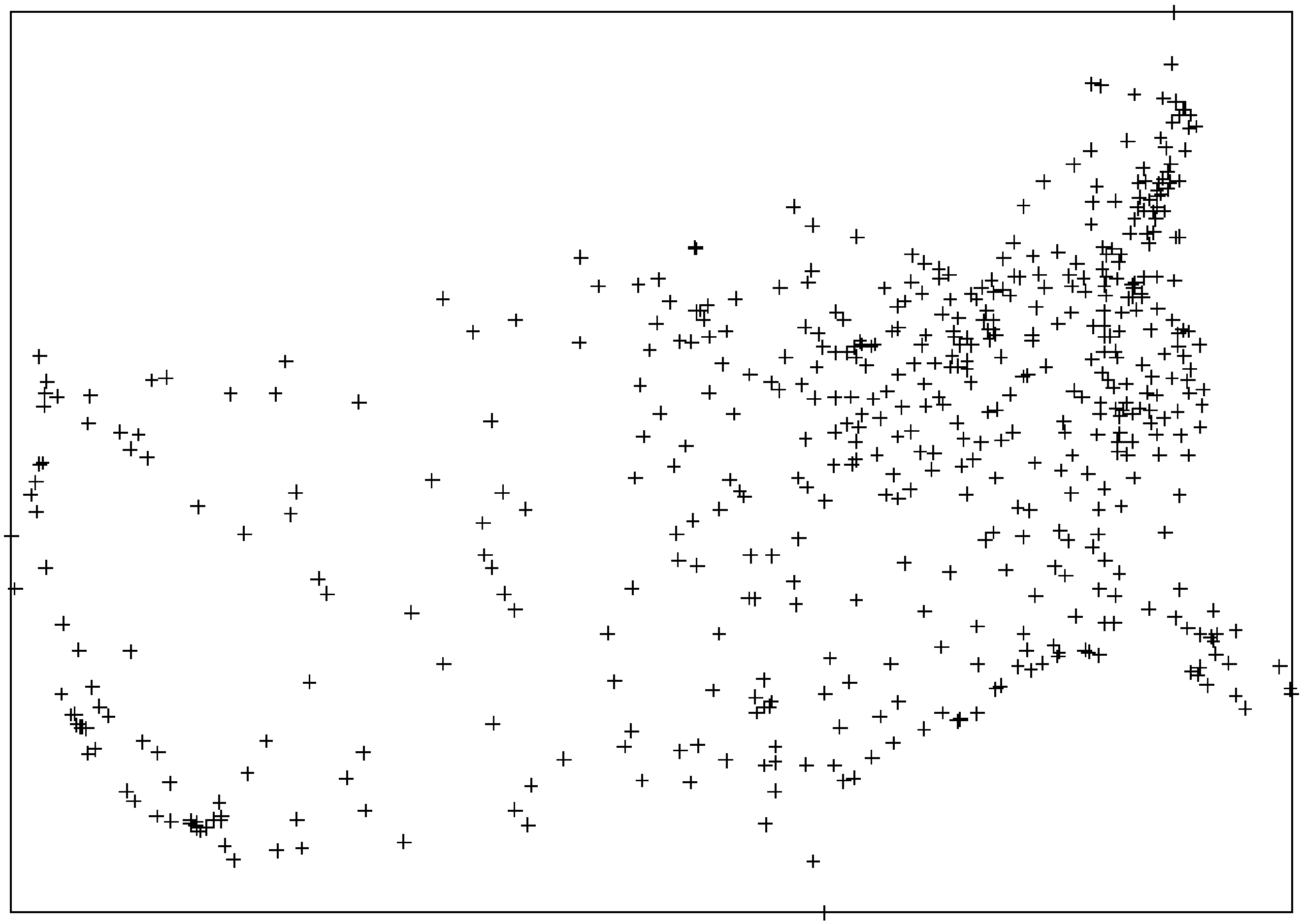
$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)}.$$

Traveling Salesman Problem

- We try to highlight differences in performance among ACO algorithms by running computational experiments on instances available from the **TSPLIB benchmark library**, which is accessible on the Web
- TSPLIB instances have been used in a number of influential studies of the TSP
- Most of the TSPLIB instances are geometric TSP instances, that is, they are defined by the coordinates of a set of points and the distance between these points is computed

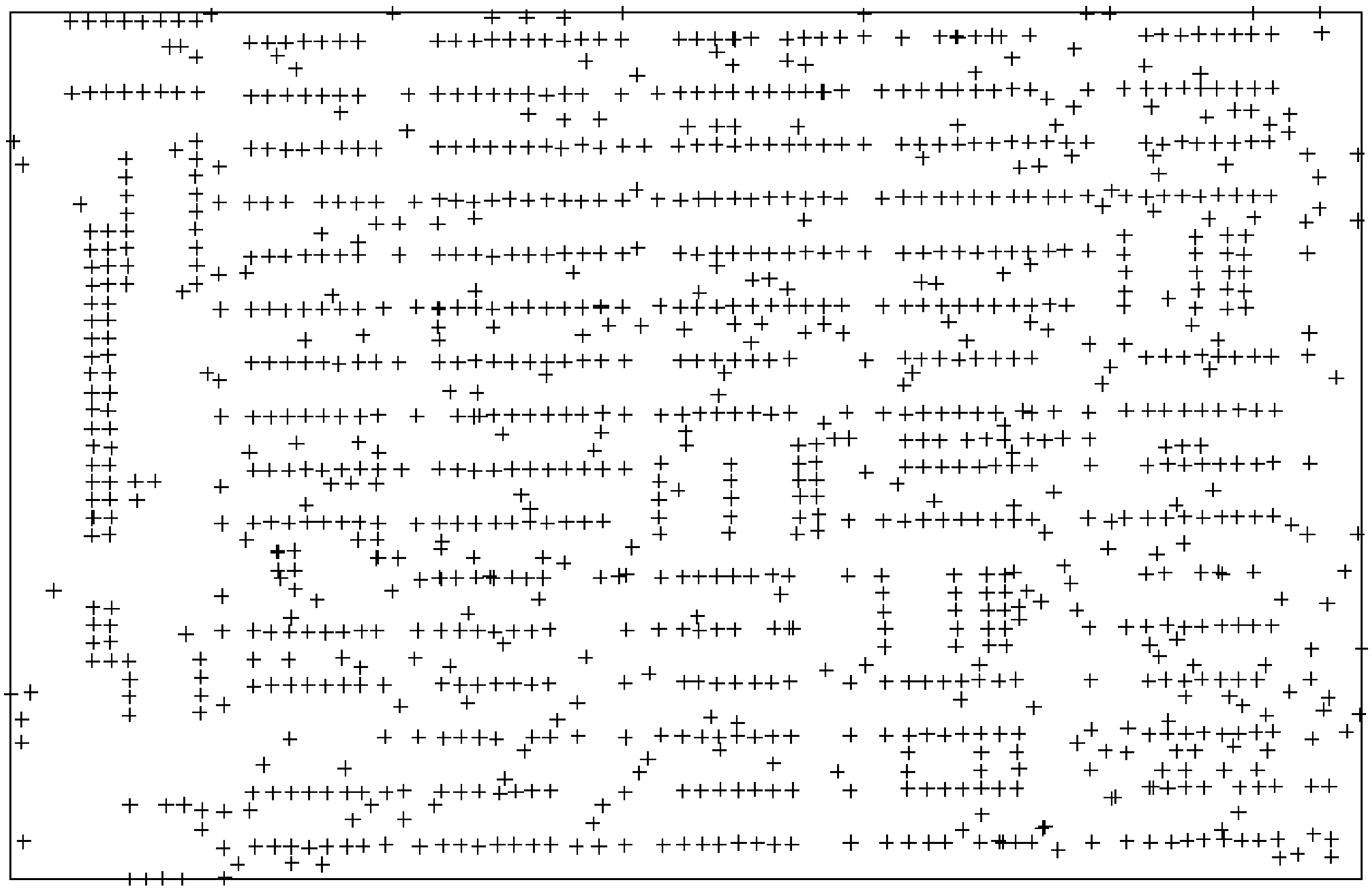
Traveling Salesman Problem

- This figure shows the TSP instance **att532**, which comprises 532 cities in the United States.



Traveling Salesman Problem

- This figure shows instance **pcb1173**, which represents the location of 1173 holes to be drilled on a printed circuit board.



ACO Algorithms for TSP

ACO Algorithms for TSP

- The construction graph $G_C = (C, L)$, where the set L fully connects the components C, is identical to the problem graph, that is $C = N$ and $L = A$
- Each connection has a weight which corresponds to the distance d_{ij} between nodes i and j.
- The states of the problem are the set of all possible tours.

ACO Algorithms for TSP

- The pheromone trails are associated with arcs and therefore τ_{ij} in the TSP refer to the desirability of visiting city *j* directly after *i*.
- The heuristic information η_{ij} is typically inversely proportional to the distance between cities *i* and *j*, a straightforward choice being $\eta_{ij} = 1/d_{ij}$.
- In fact, this is also the heuristic information used in most ACO algorithms for the TSP.

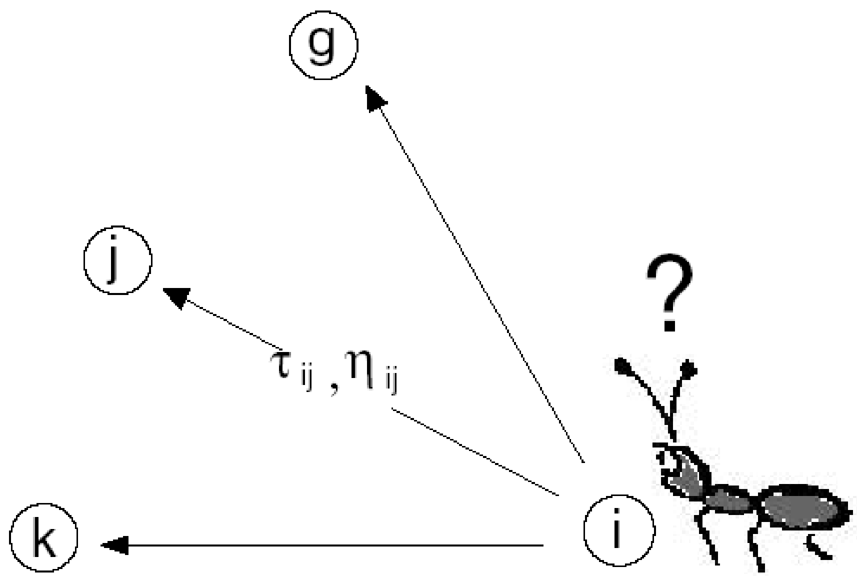
ACO Algorithms for TSP

- Tours are constructed by applying the following simple constructive procedure to each ant:

1. choose, according to some criterion, a start city at which the ant is positioned;
2. use pheromone and heuristic values to probabilistically construct a tour by iteratively adding cities that the ant has not visited yet, until all cities have been visited; and
3. go back to the initial city.

Pheromone trails and heuristic information

- The probabilistic decision is a function of pheromone trails and heuristic values



- After all ants have completed their tour, they may deposit pheromone on the tours they have followed.

ACO Algorithmic scheme for TSP

- **The algorithmic scheme**:

```
procedure ACOMetaheuristicStatic
    Set parameters, initialize pheromone trails
    while (termination condition not met) do
        ConstructAntsSolutions
        ApplyLocalSearch          % optional
        UpdatePheromones
    end
end
```

ACO Algorithmic scheme for TSP

- After initializing the parameters and the pheromone trails, these ACO algorithms iterate through a main loop
- **First** all of the ants' tours are constructed
- **Then** an optional phase takes place in which the ants' tours are improved by the application of some **local search algorithm**
- **Finally** the pheromone trails are updated.
  - This last step involves pheromone evaporation and the update of the pheromone trails by the ants to reflect their search experience.

ACO Algorithmic scheme for TSP

- The application of a **local search algorithm** is a typical example of a possible **daemon action** in ACO algorithms.
- We will see that, in some cases, before adding pheromone, the tours constructed by the ants may be improved by the application of a local search procedure.

ACO Algorithms for TSP

- The first ACO algorithm, **Ant System (AS)**, was introduced using the TSP as an example application.
- AS achieved encouraging initial results
- The extensions of AS that significantly improved performance:
  - **Elitist AS**
  - **Rank-based AS**
  - **MAX–MIN AS**

ACO Algorithms for TSP

- Similarity between AS and of these extensions:
  - the same **solution construction procedure**
  - the same **pheromone evaporation procedure**
- The main differences between AS and these extensions are:
  - the way **the pheromone update** is performed
  - some additional details in the **management of the pheromone trails**

ACO Algorithms for TSP

- A few other ACO algorithms that more substantially modify the features of AS were also proposed in the literature.
- These extensions include:
  - **Ant-Q**
  - **Ant Colony System (ACS)**
  - **Approximate Nondeterministic Tree Search (ANTS)**
  - **Hyper-cube framework for ACO**

ACO Algorithms

- We note that not all available ACO algorithms have been applied to the TSP
- Exceptions are:
  - **ANTS algorithm**
  - **Hyper-cube framework**

ACO Algorithms

ACO algorithm	TSP	Main references
Ant System (AS)	yes	Dorigo (1992); Dorigo, Maniezzo, & Colorni (1991a,b, 1996)
Elitist AS	yes	Dorigo (1992); Dorigo, Maniezzo, & Colorni (1991a,b, 1996)
Ant-Q	yes	Gambardella & Dorigo (1995); Dorigo & Gambardella (1996)
Ant Colony System	yes	Dorigo & Gambardella (1997a,b)
$\mathcal{MAX}$–$\mathcal{MIN}$ AS	yes	Stützle & Hoos (1996, 2000); Stützle (1999)
Rank-based AS	yes	Bullnheimer, Hartl, & Strauss (1997, 1999c)
ANTS	no	Maniezzo (1999)
Hyper-cube AS	no	Blum, Roli, & Dorigo (2001); Blum & Dorigo (2004)

Ant System

Ant System

- Initially, three different versions of AS were proposed:
  - **Ant-density**
  - **Ant-quantity**
  - **Ant-cycle**
- In the **ant-density** and **ant-quantity** versions the ants updated the pheromone directly after a move from one city to an adjacent city

Ant System

- In the **ant-cycle** version the pheromone update was only done **after all the ants** had constructed the tours and the amount of pheromone deposited by each ant was set to be a function of the tour quality.
- Nowadays, when referring to AS, one actually refers to **ant-cycle** since the two other variants were **abandoned** because of their **low-quality performance**.
- The two main phases of the AS algorithm:
  - the ants' solution construction and
  - the pheromone update

Initialization the Pheromone Trails

- **If the initial pheromone values are too low**
  - then the search is quickly biased by the first tours generated by the ants, which in general leads toward the exploration of lesser zones of the search space.
- **If the initial pheromone values are too high**
  - then many iterations are lost waiting until pheromone evaporation reduces enough pheromone values, so that pheromone added by ants can start to bias the search.

Initialization the Pheromone Trails

- In AS the pheromone trails is to set them to a value slightly **higher than** the expected amount of pheromone deposited by the ants in one iteration
- A rough estimate of this value can be obtained by setting, , $\tau_{ij} = \tau_0 = m/C^{nn}$,
  - where m is the number of ants, and
  - C^{nn} is the length of a tour generated by **the nearest-neighbor heuristic**

Tour Construction

- In AS, m (artificial) ants **concurrently** build a tour of the TSP.
- Initially, ants are put on randomly chosen cities.
- At each construction step, ant k applies a probabilistic action choice rule, called random proportional rule, to decide which city to visit next.

Tour Construction

- In particular, the probability with which ant k, currently at city i, chooses to go to city j is:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \quad \text{if } j \in \mathcal{N}_i^k,$$

  - $\eta_{ij}=1/d_{ij}$ is a heuristic value
  - α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information
  - N_i^k is the feasible neighborhood of ant k when being at city i, that is, the set of cities that ant k has not visited yet (the probability of choosing a city outside N_i^k is 0).

Tour Construction

- **If $\alpha = 0$,**
  - the closest cities are more likely to be selected: this corresponds to a classic stochastic greedy algorithm.
- **If $\beta = 0$,**
  - only pheromone is used, without any heuristic bias.
  - This generally leads to rather poor results
- **If $\alpha > 1$,**
  - it leads to the rapid emergence of a situation in which all the ants follow the same path and construct the same tour, which, in general, is strongly suboptimal

Tour Construction

- Good parameter values for the AS are:
  - **The parameter α:** $\alpha = 1$
  - **The parameter β:** $\beta = 2$ to 5
  - **Evaporation rate:** $\rho = 0.5$
  - **The number of ants:** $m = n$ (the number of cities)
  - **The initialization value of pheromone trial:** $\tau_0 = m/C^{nn}$
- It should be clear that in individual instances, different settings may result in much better performance. However, these parameters were found to yield reasonable performance over a significant set of TSP instances.

Tour Construction

- Each ant k maintains a memory M^k which contains the cities already visited, in the order they were visited.
- This memory is used:
  - to define the feasible neighborhood
  - to compute the length of the tour T^k it generated and
  - to retrace the path to deposit pheromone.

Tour Construction

- There are two different ways of implementing solution construction:
  - **Parallel solution construction**: at each construction step all the ants move from their current city to the next one
  - **Sequential implementation**: an ant builds a complete tour before the next one starts to build another one.
- Both choices for the implementation of the tour construction are equivalent in the sense that they do not significantly influence the algorithm's behavior.

Update of Pheromone Trails

- After all the ants have constructed their tours, the pheromone trails are updated.
- This is done by:
  - **first** lowering the pheromone value on all arcs by a constant factor, and
  - **then** adding pheromone on the arcs the ants have crossed in their tours.

Update of Pheromone Trails

- Pheromone evaporation is implemented by:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \quad \forall (i, j) \in L,$$

  – where $0 < \rho \leq 1$ is the pheromone evaporation rate.
- The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and it enables the algorithm to ''forget'' bad decisions previously taken.
- In fact, if an arc is not chosen by the ants, its associated pheromone value decreases exponentially in the number of iterations.

Update of Pheromone Trails

- After evaporation, all ants deposit pheromone on the arcs they have crossed in their tour:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \quad \forall (i,j) \in L,$$

- where $\Delta\tau^{k}_{ij}$ is the amount of pheromone ant k deposits on the arcs it has visited.

Update of Pheromone Trails

- $\Delta\tau^{k}_{ij}$ is defined as follows:

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C^{k}, & \text{if arc } (i,j) \text{ belongs to } T^{k}; \\ 0, & \text{otherwise;} \end{cases}$$

- where C^k is the **length of the tour** T^k built by the k-th ant, is computed as the sum of the lengths of the arcs belonging to T^k.
- The better an ant's tour is, the more pheromone the arcs belonging to this tour receive.

Update of Pheromone Trails

- In general, arcs that are used by many ants and which are part of short tours, receive more pheromone and are therefore more likely to be chosen by ants in future iterations of the algorithm.
- As we said, the relative performance of AS when compared to other Metaheuristics tends to decrease dramatically as the size of the test-instance increases.
- Therefore, a substantial amount of research on ACO has focused on how to improve AS.

Elitist Ant System

Elitist Ant System

- A first improvement on the initial AS, called the **elitist strategy** for Ant System **(EAS)**
- It was introduced in Dorigo (1992) and Dorigo et al., (1991a, 1996).
- The idea is to provide strong additional reinforcement to the arcs belonging to the **best tour** found since the start of the algorithm
- This tour is denoted as $\mathbf{T^{bs}}$ (**best-so-far tour**)
- An additional pheromone deposited by an additional ant called **best-so-far ant**

Update of Pheromone Trails

- The additional reinforcement of tour T^{bs} is achieved by adding a quantity e / C^{bs} to its arcs, where e is a parameter that defines the weight given to the best-so-far tour T^{bs}, and C^{bs} is its length.
- Thus equation for the pheromone deposit becomes:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs}, \quad \forall (i, j) \in L,$$

- Note that this additional feedback to the best-so-far tour is an example of a **daemon action** of the ACO metaheuristic.

Update of Pheromone Trails

- Where $\Delta\tau^{k}_{ij}$ and $\Delta\tau^{bs}_{ij}$ are defined as follows:

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C^{k}, & \text{if arc } (i, j) \text{ belongs to } T^{k}; \\ 0, & \text{otherwise;} \end{cases}$$

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1/C^{bs}, & \text{if arc } (i, j) \text{ belongs to } T^{bs}; \\ 0, & \text{otherwise.} \end{cases}$$

Update of Pheromone Trails

- In EAS pheromone **evaporation** is implemented as in AS.
- Computational results suggest that the use of the **elitist strategy** with an appropriate value for parameter **e** allows AS to:
  - find better tours and
  - find them in a lower number of iterations

Parameter Values

- Good parameter values for the EAS are:
  - **The parameter α:** $\alpha = 1$
  - **The parameter β:** $\beta = 2$ to 5
  - **Evaporation rate:** $\rho = 0.5$
  - **The number of ants:** $m = n$ (the number of cities)
  - **The initialization value of pheromone trial:** $\tau_0 = (e + m) / \rho C^{nn}$
  - **The parameter e:** $e = n$

Rank-Based Ant System

Rank-Based Ant System

- Another improvement over AS is the **rank-based version of AS** (AS_{rank})
- It was proposed by **Bullnheimer** et al. (1999c).
- In AS_{rank} each ant deposits an amount of pheromone that decreases with its rank.
- Additionally, as in EAS, **the best-so-far ant** always deposits the largest amount of pheromone in each iteration.

Update of Pheromone Trails

- Before updating the pheromone trails, the ants are sorted by increasing tour length
- The quantity of pheromone an ant deposits is weighted according to the **rank r** of the ant
- Ties can be solved randomly.
- In each iteration only the **(w – 1) best-ranked ants** and the ant that produced **the best-so-far tour** are allowed to deposit pheromone.
- The ant that produced **the best-so-far tour** does not necessarily belong to the set of ants of the current algorithm iteration.

Update of Pheromone Trails

- **The best-so-far tour** gives the strongest feedback, with weight w
- Its contribution $1 / C^{bs}$ is multiplied by w
- The **r-th best ant** of the current iteration contributes to pheromone updating with the value $1 / C^{r}$ multiplied by a weight given by $\max\{0; w - r\}$.

Update of Pheromone Trails

- Thus, the AS_{rank} pheromone update rule is:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{r=1}^{w-1} (w - r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs}, \quad \forall (i, j) \in L,$$

- where = 1 / C^r and = 1 / C^{bs}

$$\Delta\tau_{ij}^{r} = \begin{cases} 1/C^{r}, & \text{if arc } (i, j) \text{ belongs to } T^{r}; \\ 0, & \text{otherwise;} \end{cases}$$

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1/C^{bs}, & \text{if arc } (i, j) \text{ belongs to } T^{bs}; \\ 0, & \text{otherwise.} \end{cases}$$

Update of Pheromone Trails

- The results of an experimental evaluation suggest that AS_{rank} performs slightly better than EAS and significantly better than AS.

Parameter Values

- Good parameter values for the **Rank-Based Ant System** are:
  - **The parameter α:** $\alpha = 1$
  - **The parameter β:** $\beta = 2$ to 5
  - **Evaporation rate:** $\rho = 0.1$
  - **The number of ants:** m = n (the number of cities)
  - **The initialization value of pheromone trial:** $\tau_0 = 0.5\, r\, (r - 1) / \rho\, C^{nn}$
  - **The parameter e:** e = n
  - **The number of ants that deposit pheromones:** w = 6

MAX–MIN Ant System

MAX–MIN Ant System

- **MAX–MIN Ant System (MMAS)** introduces four main modifications with respect to AS.
- It was introduced by Stützle & Hoos (1997, 2000); Stützle, (1999).

MAX–MIN Ant System

- **First modification:**
  - only either the **iteration-best ant**, that is, the ant that produced the best tour in the current iteration, or **the best-so-far ant** is allowed to deposit pheromone.
  - the first modification may lead to a stagnation situation in which all the ants follow the **same tour**, because of the excessive growth of pheromone trails on arcs of a good, although **suboptimal**, tour.
  - To counteract this effect, a second modification introduced by MMAS.
- **Second modification**:
  - It limits the possible range of pheromone trail values to the interval $[\tau_{min}, \tau_{max}]$.

MAX–MIN Ant System

- **Third modification:**
  - the pheromone trails **are initialized** to the **upper pheromone trail limit**
  - which, together with a small pheromone evaporation rate, increases the exploration of tours at the start of the search.
- **Fourth modification:**
  - The pheromone trails **are reinitialized** each time the system approaches stagnation or when no improved tour has been generated for a certain number of consecutive iterations.

Update of Pheromone Trails

- After all ants have constructed a tour, pheromones are updated by applying evaporation as in AS, followed by the deposit of new pheromone as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \quad \forall (i,j) \in L,$$

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best},$$

- where $\Delta\tau^{best}_{ij} = 1 / C^{best}$.

Update of Pheromone Trails

- The ant which is allowed to add pheromone may be either:
  - the best-so-far, in which case $\Delta\tau^{best}_{ij} = 1 / C^{bs}$ or
  - the iteration-best, in which case $\Delta\tau^{best}_{ij} = 1 / C^{ib}$, where C^{ib} is the length of the iteration-best tour.
- In MMAS implementations both the iteration-best and the best-so-far update rules are used, in an alternate way.

Update of Pheromone Trails

- The choice of the relative frequency with which the two pheromone update rules are applied has an influence on how greedy the search is:
  - When pheromone updates are always performed by the **best-so-far ant**, the search focuses very quickly around T^{bs}
  - when it is the **iteration-best ant** that updates pheromones, then the number of arcs that receive pheromone is larger and the search is less directed.

Update of Pheromone Trails

- Experimental results indicate that:
  - for **small TSP instances** it may be best to use only **iteration-best pheromone updates**
  - for **large TSPs** with several hundreds of cities the best performance is obtained by giving an increasingly stronger emphasis to the **best-so-far tour**
- This can be achieved, for example, by gradually increasing the frequency with which the best-so-far tour T^{bs} is chosen for the trail update.

Trails Pheromone Trail Limits

- In MMAS, lower and upper limits τ_{min} and τ_{max} on the possible pheromone values on **any arc** are imposed in order to avoid search stagnation.
- The imposed pheromone trail limits have the effect of limiting the probability p_{ij} of selecting a city j when an ant is in city i to the interval $[p_{min}, p_{max}]$, with
  $\mathbf{0 < p_{min} \leq p_{ij} \leq p_{max} \leq 1}$
- Only when an ant k has just one single possible choice for the next city, that is $|N_i^k|=1$, we have $p_{min} = p_{max}=1$

Trails Pheromone Trail Limits

- MMAS uses an estimate of this value, $1 / \rho C^{bs}$, to define τ_{max}: each time a **new best-so-far tour** is found, the value of τ_{max} is updated.
- The lower pheromone trail limit is set to $\tau_{min} = \tau_{max} / a$,
  - where a is a parameter
- Experimental results suggest that, in order to avoid stagnation, τ_{min} play a more important role than τ_{max}.

Pheromone Trail Initialization and Reinitialization

- At the start of the algorithm, the **initial pheromone trails** are set to an estimate of the upper pheromone trail limit.
- A small pheromone evaporation parameter causes a slow increase in the relative difference in the pheromone trail levels, so that the initial search phase of MMAS is very explorative.
- As a further means of increasing the exploration of paths that have only a small probability of being chosen, in MMAS pheromone trails are occasionally reinitialized.

Pheromone Trail Initialization and Reinitialization

- **Pheromone trail reinitialization** is typically triggered when the algorithm approaches the stagnation behavior
  - e.g. if for a given number of algorithm iterations no improved tour is found.

Parameter Values

- Good parameter values for the **MMAS** are:
  - **The parameter α:** $\alpha = 1$
  - **The parameter β:** $\beta = 2$ to 5
  - **Evaporation rate:** $\rho = 0.02$
  - **The number of ants:** $m = n$ (the number of cities)
  - **The initialization value of pheromone trial:** $\tau_0 = 1 / \rho C^{nn}$
  - **The pheromone trail limits are:**

$$\tau_{max} = 1/\rho C^{bs}$$

$$\tau_{min} = \tau_{max}(1 - \sqrt[n]{0.05})/((avg - 1) \cdot \sqrt[n]{0.05})$$

    - where **avg** is the average number of different choices available to an ant at each step while constructing a solution

MAX–MIN Ant System

- MMAS is one of **the most studied** ACO algorithms and it has been extended in many ways.
- In one of these extensions, the pheromone update rule occasionally uses the best tour found since the most recent reinitialization of the pheromone trails instead of the best-so-far tour.

Ant Colony System

Ant Colony System

- **Ant Colony System (ACS)** introduced by Dorigo & Gambardella, 1997a,b.

Ant Colony System

- ACS differs from AS in three main points.
  - **First,** It exploits the search experience accumulated by the ants more strongly than AS does through the use of a more **aggressive action choice rule**.
  - **Second,** Pheromone evaporation and pheromone deposit take place only on the arcs belonging to the **best-so-far tour**.
  - **Third,** Each time an ant uses an arc (i, j) to move from city i to city j, it removes some pheromone from the arc to increase the exploration of alternative paths.

Tour Construction

- In ACS, when located at city i, ant k moves to a city j chosen according to the socalled **pseudorandom proportional rule**, given by

$$j = \begin{cases} \operatorname{argmax}_{l \in \mathcal{N}_i^k} \{\tau_{il}[\eta_{il}]^\beta\}, & \text{if } q \leq q_0; \\ J, & \text{otherwise;} \end{cases}$$

  - where q is a random variable uniformly distributed in [0, 1]
  - q_0 ($0 \leq q_0 \leq 1$) is a parameter
  - J is calculated by ($\alpha = 1$):

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \quad \text{if } j \in \mathcal{N}_i^k$$

Tour Construction

- **With probability q_0**
  - the ant makes the best possible move as indicated by the **learned pheromone trails and the heuristic information**
  - In this case, the ant is **exploiting** the learned knowledge
  - It concentrates the search of the system around the **best-so-far solution** or to explore other tours.
- **With probability $1 - q_0$**
  - It performs a biased **exploration** of the arcs.

Global Pheromone Trail Update

- In ACS **only one ant** (the best-so-far ant) is allowed to add pheromone after each iteration.
- The update in ACS is implemented by the following equation:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \quad \forall(i,j) \in T^{bs}$$

  - where $\Delta\tau^{best}_{ij} = 1 / C^{bs}$
- Both evaporation and new pheromone deposit, only applies to the arcs of T^{bs}, not to all the arcs as in AS.
- In this way the computational complexity of the pheromone update at each iteration is reduced

Global Pheromone Trail Update

- In initial experiments, the use of the **iteration-best tour** was also considered for the pheromone updates,
  - for small TSP instances the differences in the final tour quality obtained by updating the pheromones using the **best-so-far** or the **iteration-best tour** was found to be minimal,
  - for instances with more than 100 cities the use of the **best-so-far tour** gave far better results

Local Pheromone Trail Update

- In addition to the global pheromone trail updating rule, in ACS the ants use a **local pheromone update rule**
- They apply immediately after having crossed an arc (i, j) during the tour construction:

  $$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\tau_0$$

  - where ζ, $0 < \zeta < 1$, is a parameter

Local Pheromone Trail Update

- The effect of the local updating rule is that each time an ant uses an arc (i, j) its pheromone trail τ_{ij} is reduced, so that the arc becomes less desirable for the following ants.
- This allows an increase in the exploration of arcs that have not been visited yet and, in practice, has the effect that the algorithm does not show a stagnation behavior (i.e., ants do not converge to the generation of a common path)

Local Pheromone Trail Update

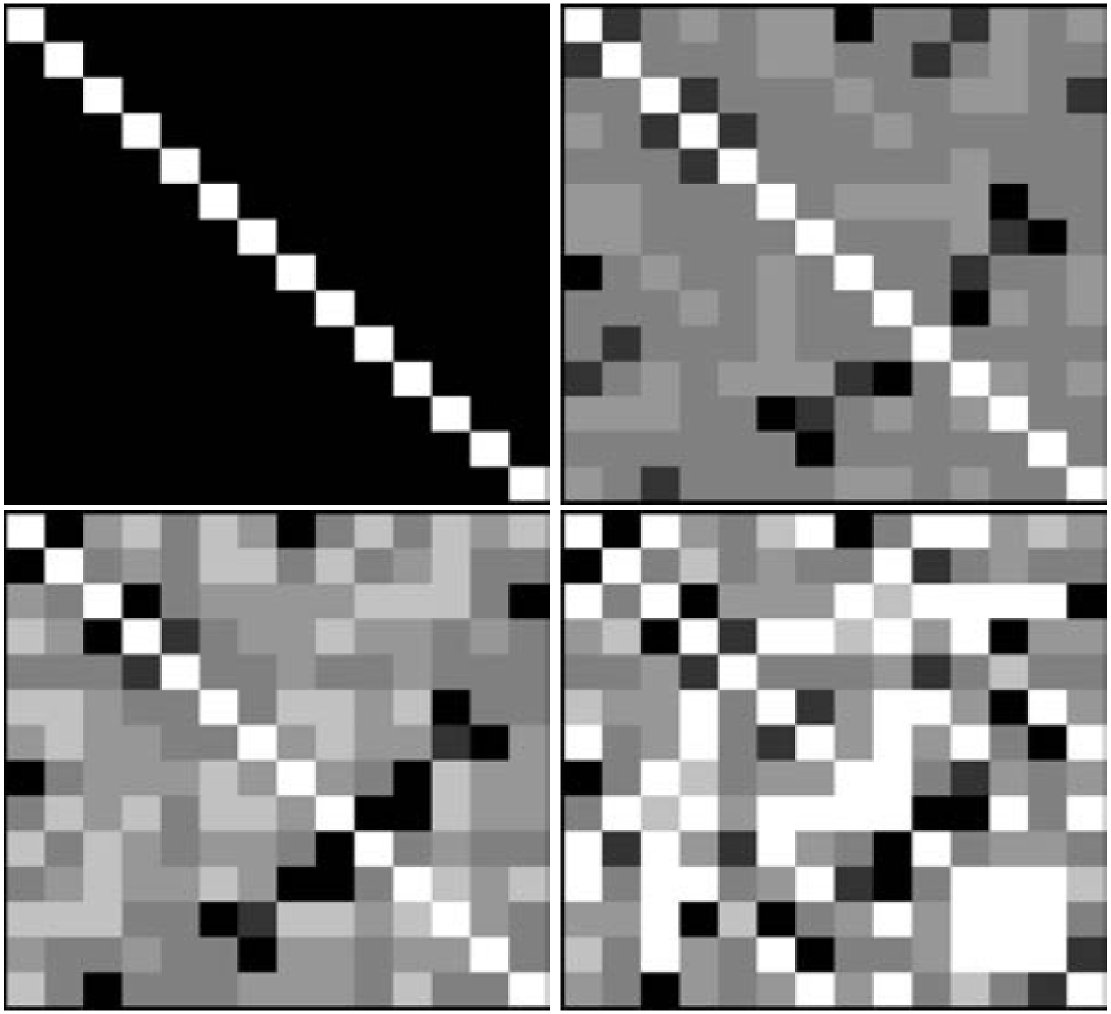
- It is important to note that, while for the previously discussed AS variants it does not matter whether the ants construct the tours in parallel or sequentially
- This makes a difference in ACS because of the **local pheromone update rule**.

Parameter Values

- Good parameter values for the **ACS** are:
  - **The parameter α:** $\alpha = 1$
  - **The parameter β:** $\beta = 2$ to 5
  - **Evaporation rate:** $\rho = 0.1$
  - **The number of ants:** $m = 10$
  - **The initialization value of pheromone trial:** $\tau_0 = 1 / nC^{nn}$
  - **The local pheromone trail update rule:** $\zeta = 0.1$
  - **The pseudorandom proportional action choice rule:** $q_0 = 0.9$

Search Stagnation

A visual representation of the pheromone matrix



Search Stagnation

- The pheromone values on the arcs, stored in the pheromone matrix, are translated into gray-scale values;
- The darker an entry, the higher the associated pheromone trail value.
- The plots, from upper left to lower right, show the pheromone value for **AS** applied to TSPLIB instance **burma14** with 14 cities after 0, 5, 10, and 100 iterations.
- The **burma14** is a **symmetric** TSP instance.

Search Stagnation

- **Search stagnation**
  - is defined as the situation in which all the ants follow the same path and construct the same solution.
- With **bad parameter** settings, an **early stagnation** of the search happened
- In such an **undesirable situation** the system has stopped to explore new possibilities and no better tour is likely to be found anymore.

Search Stagnation

- Several measures may be used to detect stagnation situations.
  - **Standard deviation**
  - **Coefficient variation (CV)**
  - **Distance between tours**
  - **The average λ-branching factor**

Search Stagnation

- **Standard deviation**
  - One of the simplest possibilities is to compute the **standard deviation σ_L** of the length of the tours the ants construct after every iteration
  - If σ_L is zero, this is an indication that all the ants follow the **same path**
  - Although σ_L can go to zero also in the very unlikely case in which the ants follow **different tours** of the same length.

Search Stagnation

- **Coefficient of variation (CV)**
  - Because the standard deviation depends on the absolute values of the tour lengths, a better choice is the use of the variation coefficient, which is independent of the scale.

Coefficient of variation =

standard deviation of the tour lengths /

the average tour length

Search Stagnation

- **The distance between tours**
  - gives a better indication of the amount of exploration the ants perform.
  - In the TSP case, a way of measuring the distance dist(T, T’) between two tours T and T’ is to count the number of arcs contained in one tour but not in the other.
  - A decrease in the average distance between the ants’ tours indicates that preferred paths are appearing, and if the average distance becomes zero, then the system has entered **search stagnation**.
  - A disadvantage of this measure is that it is computationally expensive

Search Stagnation

- **The average λ-branching factor**
  - Measures the distribution of the pheromone trail values more directly.
  - If for a given city i the concentration of pheromone trail on almost all the arcs becomes very small but is large for a few others, the freedom of choice for extending partial tours from that city is very limited.
  - Consequently, if this situation arises simultaneously for all the nodes of the graph, the part of the search space that is effectively searched by the ants becomes relatively small.

Search Stagnation

- **The average λ-branching factor**
  - If τ^i_{max} is the maximal and τ^i_{min} the minimal pheromone trail value on arcs incident to node i, the λ-branching factor is given by the number of arcs incident to i that have a pheromone trail value

$$\tau_{ij} \geq \tau^i_{min} + \lambda(\tau^i_{max} - \tau^i_{min})$$

  - The value of λ ranges over the interval [0, 1]
  - **The values of the λ-branching factors** range over the interval **[2, n – 1],** where n is the number of nodes in the construction graph (which, in the TSP case, is the same as the number of cities).

Search Stagnation

- **The average of the λ-branching factors**
  - **The average of the λ-branching factors** of all nodes and gives an indication of the size of the search space effectively being explored by the ants.
  - If, for example, the average is very close to 3, on average only three arcs for each node have a high probability of being chosen.
  - Note that in the TSP the minimal average λ-branching factor is 2, because for each city there must be at least two arcs used by the ants to reach and to leave the city while building their solutions.
  - A disadvantage of the l-branching factor is that its values depend on the setting of the parameter λ.

Experimental Evaluation

Experimental Evaluation

- All the experiments were performed either on
  - 700 MHz Pentium III double-processor machine with 512 MB of RAM
  - 1.2 GHz Athlon MP double-processor machine with 1 GB of RAM
  - both machines were running SUSE Linux 7.3.

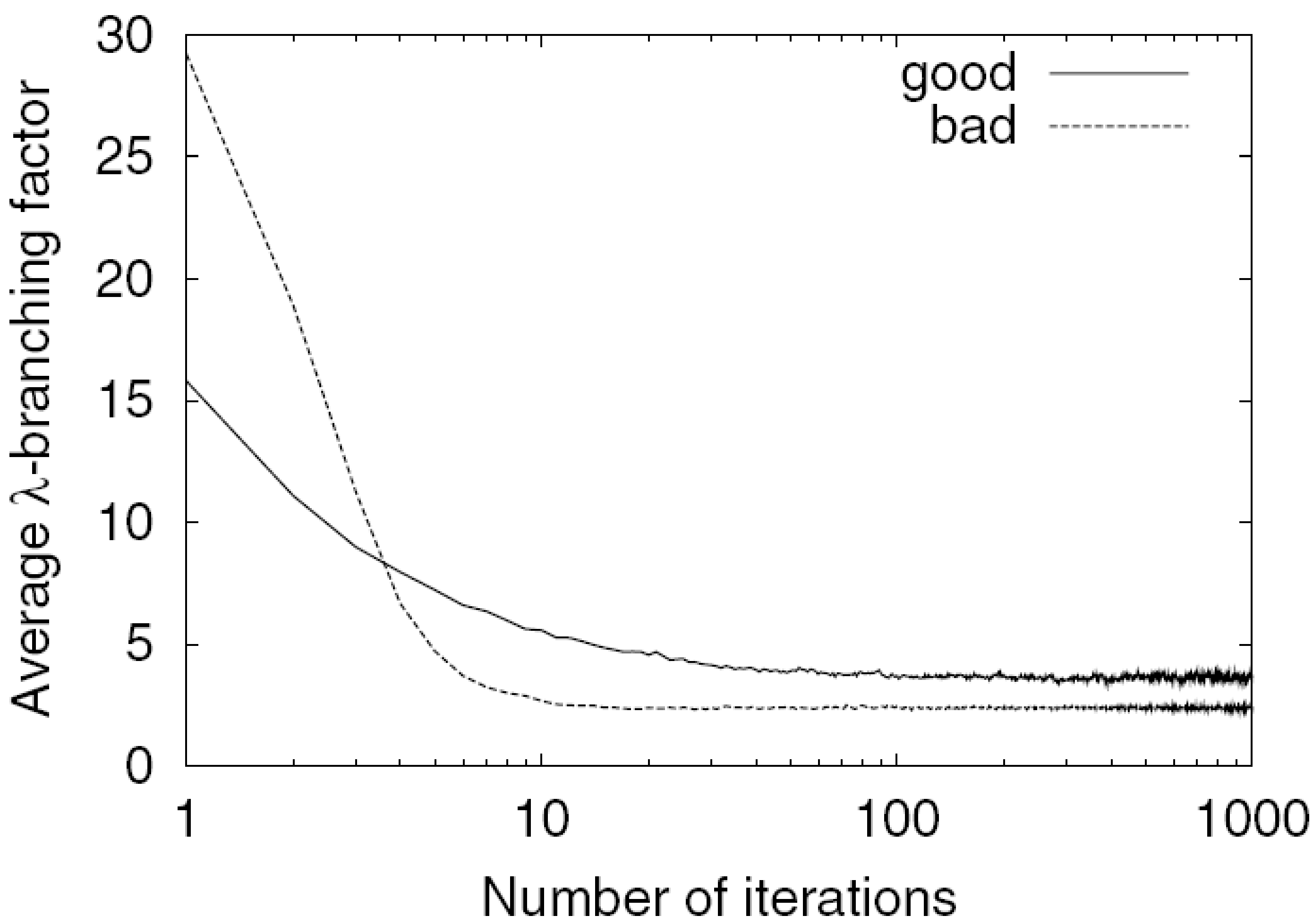
Behavior of AS

- We show the typical behavior of
  - **the average λ-branching factor** (λ **= 0.05**) and of
  - **the average distance among tours**
  - when AS has parameter settings that result in either good or bad algorithm performance.

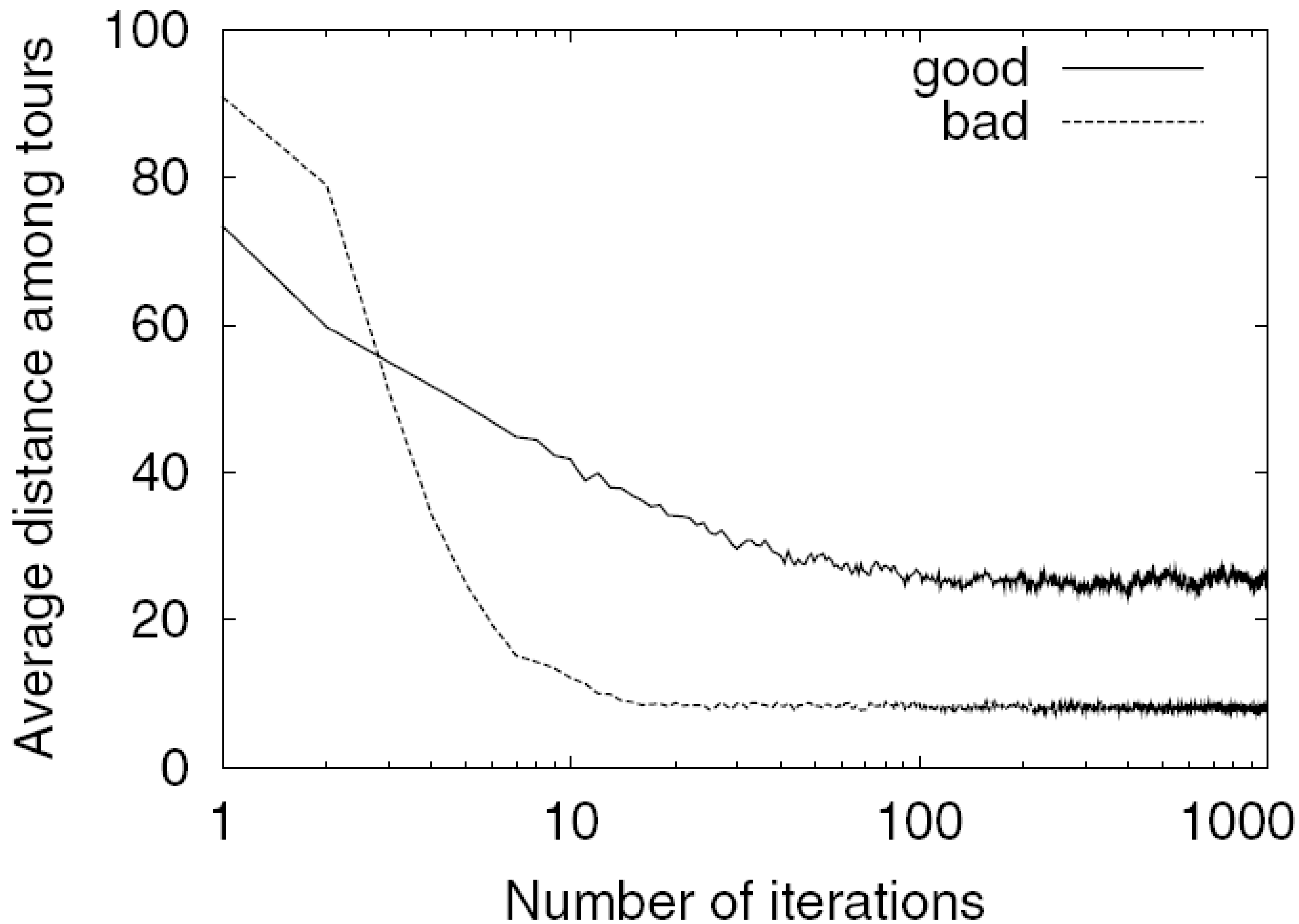
Behavior of AS

- The parameter settings are denoted by **good** and **bad** and the values used are
  - $\alpha = 1$, $\beta = 2$, $m = n$
  - $\alpha = 5$, $\beta = 0$, $m = n$
- Bad behavior because of **early stagnation**
- Example: TSPLIB instance kroA100

Behavior of AS



Behavior of AS



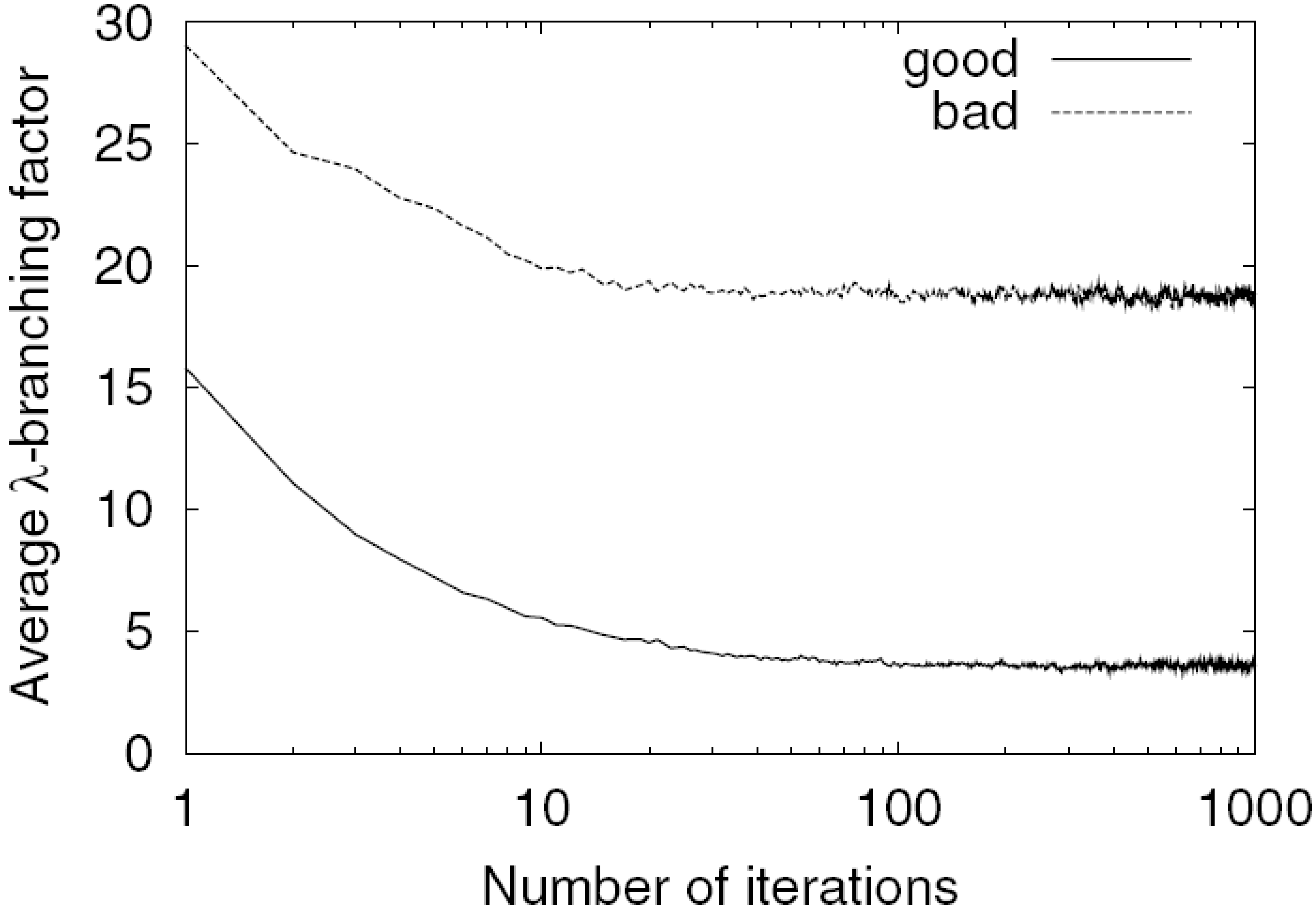
Behavior of AS

- The experimental results suggest that:
  - **if α is set to a large value,** AS enters stagnation behavior
  - **if α is chosen to be much smaller than 1,** AS does not find high-quality tours

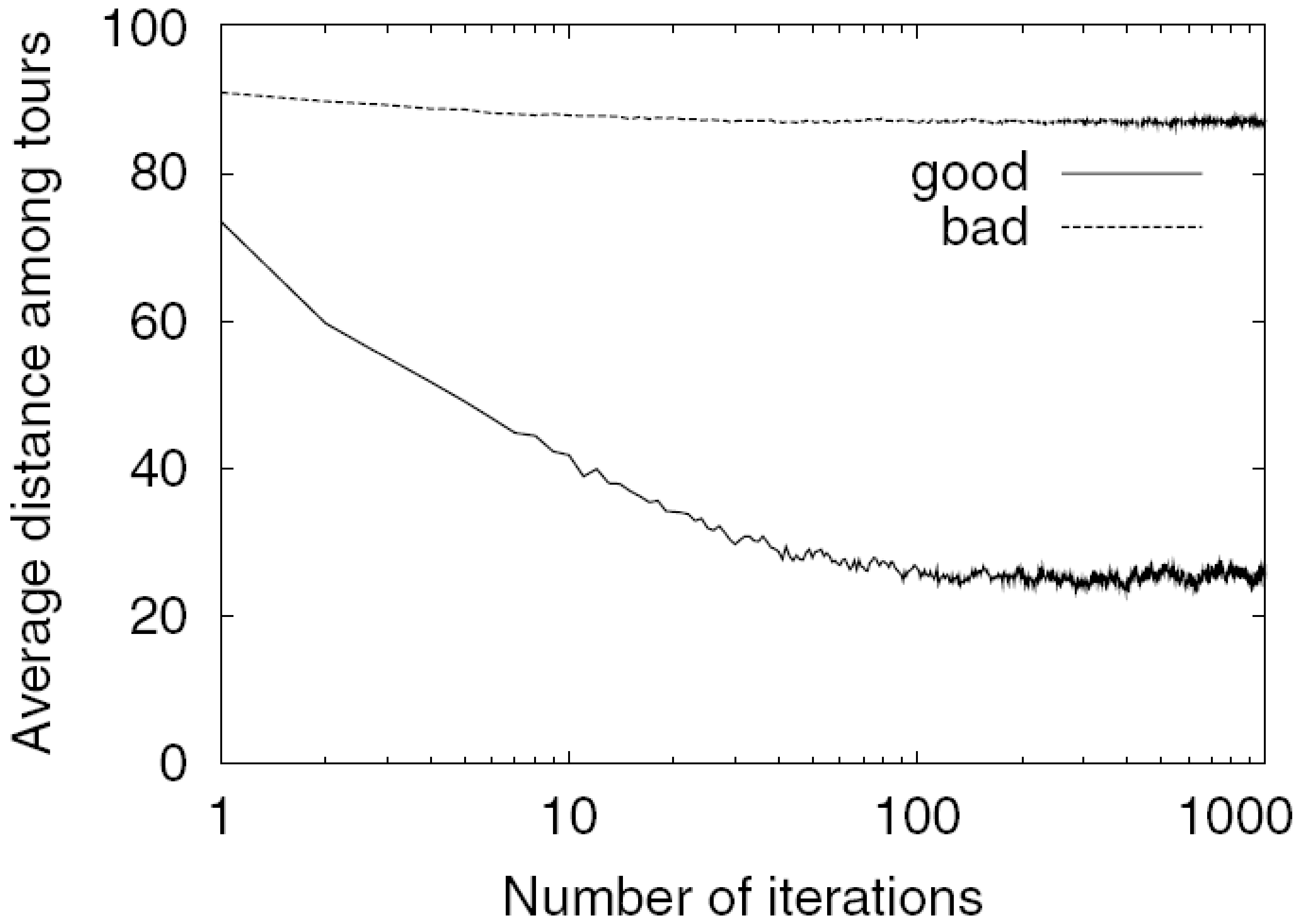
Behavior of AS

- An example of bad system behavior that occurs if **the amount of exploration is too large**
- Here, **good** refers to the same parameter setting
  - $\alpha = 1, \beta = 2, m = n$
- And bad refers to the setting
  - $\alpha = 1, \beta = 0, m = n$

Behavior of AS



Behavior of AS



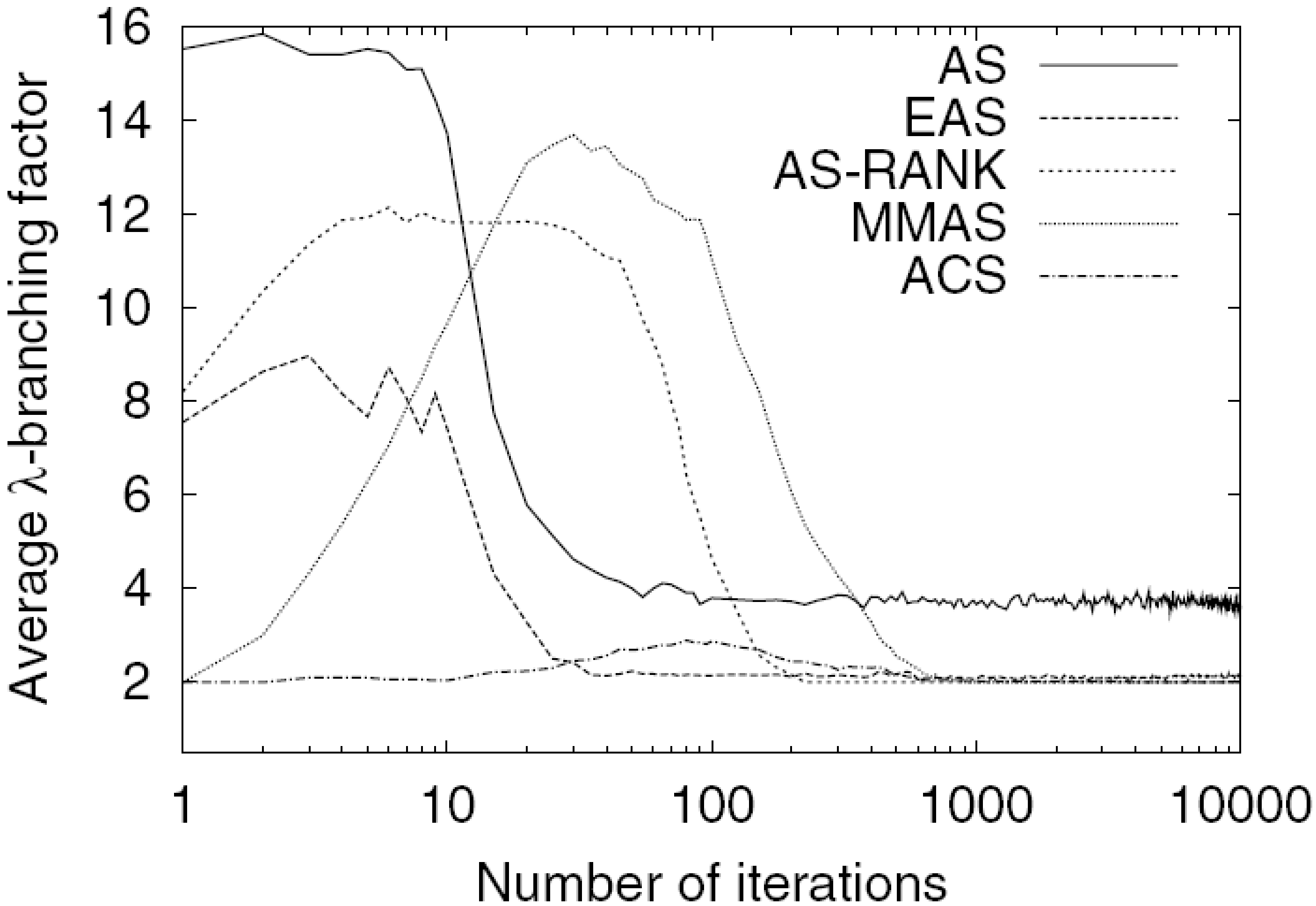
Behavior of AS

- For both stagnation measures, the algorithm using the bad parameter setting is not able to focus the search on the most promising parts of the search space.
- The overall result suggests that for AS good parameter settings are those that find a reasonable balance between a **too narrow** focus of the search process, which in the worst case may lead to stagnation behavior, and a too weak guidance of the search, which can cause excessive exploration.

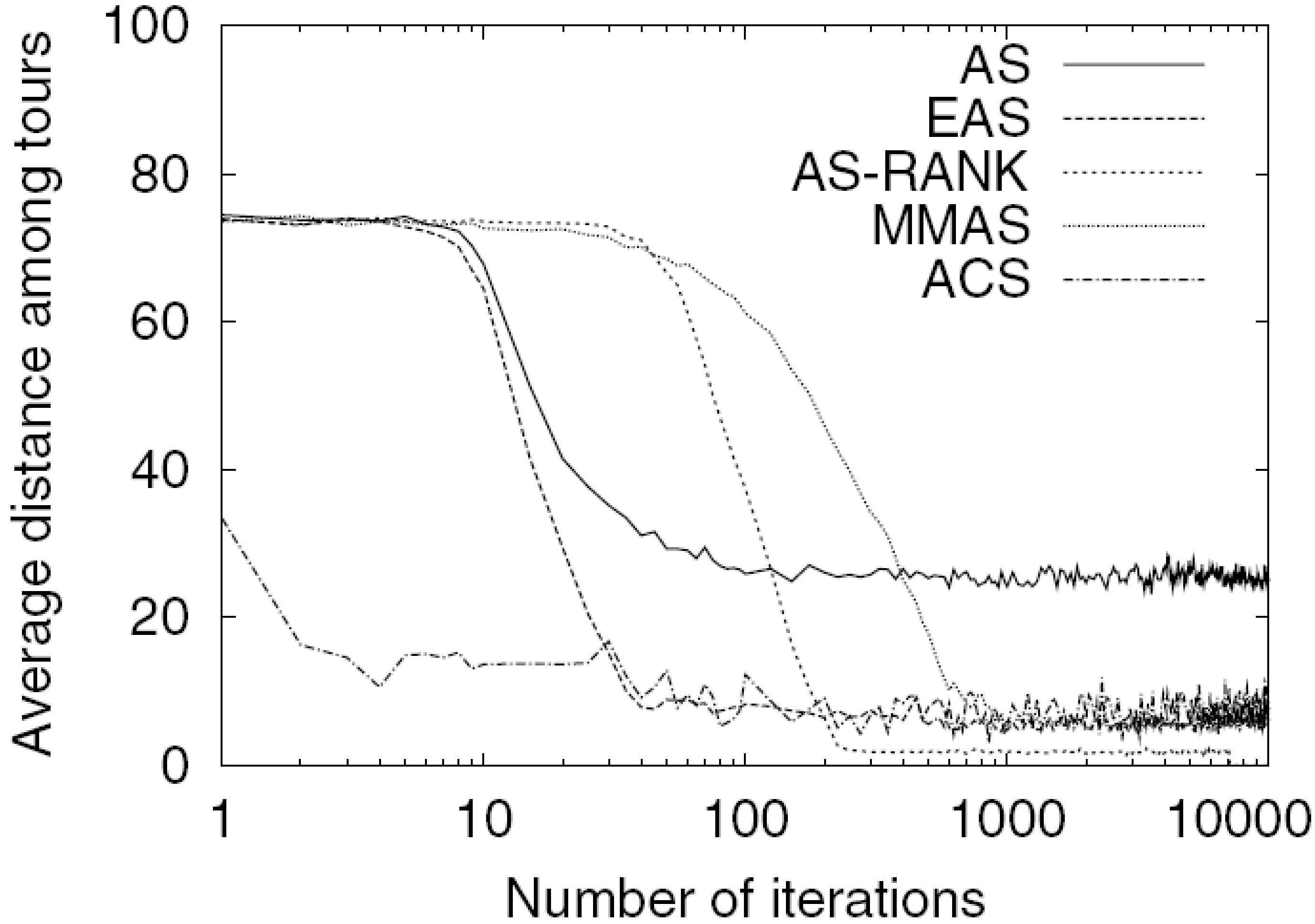
Behavior of Extensions of AS

- One particularity of AS extensions is that they direct the ants' search in a more aggressive way.
- This is mainly achieved by a stronger emphasis given to the best tours found during each iteration (e.g., in MMAS) or the best-so-far tour (e.g., in ACS).
- We would expect that this stronger focus of the search is reflected by statistical measures of the amount of exploration.

Behavior of Extensions of AS



Behavior of Extensions of AS



Behavior of Extensions of AS

- ACS shows a **low λ-branching factor** and **small average distances between the tours** throughout the algorithm's entire run
- For the others algorithms a transition from a more explorative search phase can be observed.
- This transition happens very soon in AS and AS_{rank}, it occurs only later in MMAS.

Behavior of MMAS

- MMAS has the longest explorative search phase.
- This is mainly due to the fact that pheromone trails are initialized to the initial estimate of τ_{max}, and that the evaporation rate is set to a low value ($\rho = 0.02$).
- Because of the low evaporation rate, it takes time before significant differences among the pheromone trails start to appear.
- When this happens, MMAS behavior changes from explorative search to a phase of exploitation of the experience accumulated in the form of pheromone trails.

Behavior of Extensions of AS

- In this phase, the pheromone on the arcs corresponding to the best-found tour rises up to the maximum value τ_{max}, while on all the other arcs it decreases down to the minimum value τ_{min}.
- This is reflected by an average λ-branching factor of 2.0.

Behavior of ACS

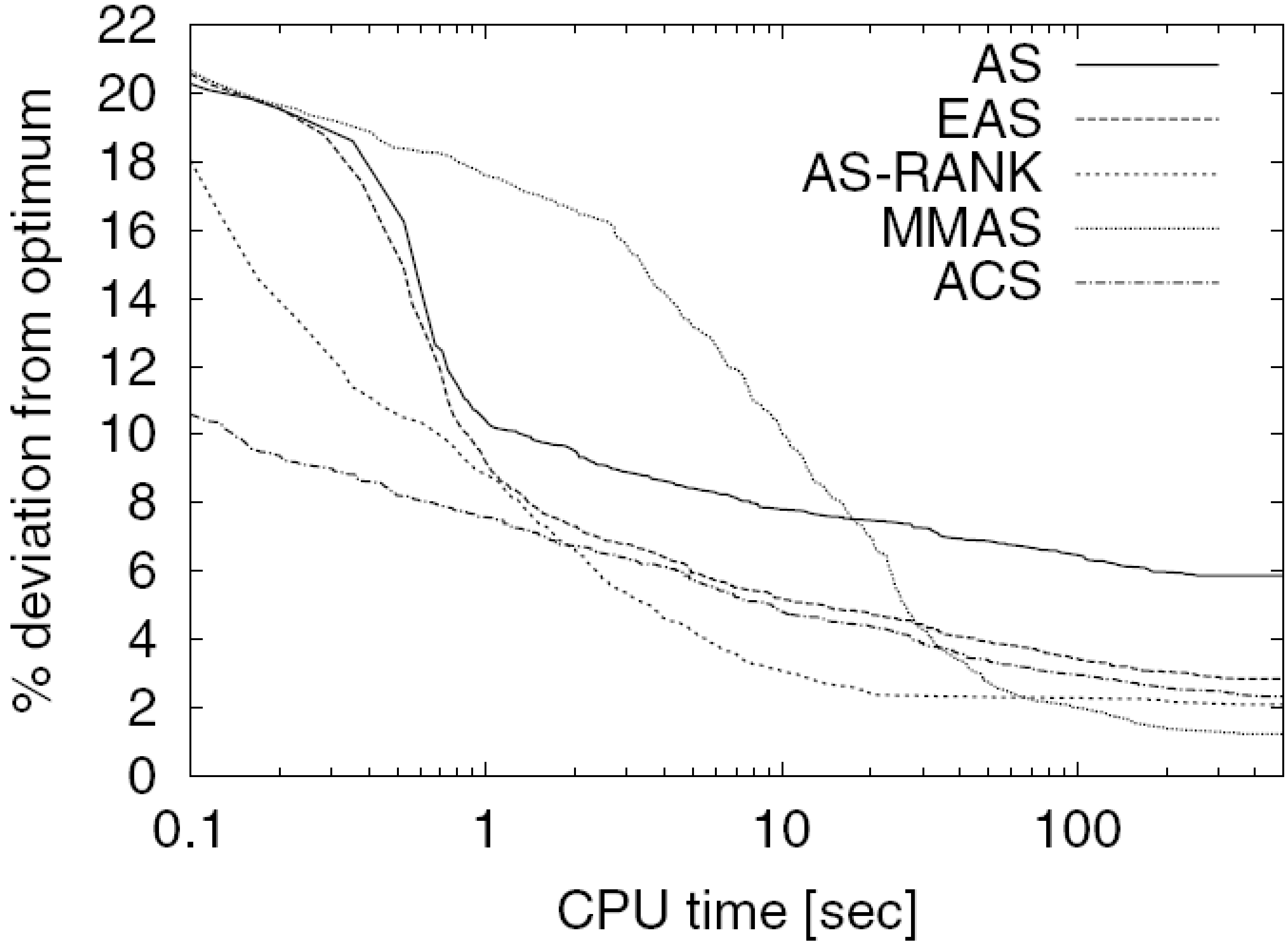
- ACS uses a very aggressive search that focuses from the very beginning around the best-so-far tour T^{bs}.
- It generates tours that differ only in a relatively small number of arcs from the best-so-far tour T^{bs}.
- This is achieved by choosing a large value for q_0 in the pseudorandom proportional action choice rule
- Local updating has the effect of lowering the pheromone on visited arcs so that they will be chosen with a lower probability by the other ants in their remaining steps for completing a tour.

Solution quality of algorithms

- We compare the development of the average solution quality measured of several algorithms as a function of the computation time.

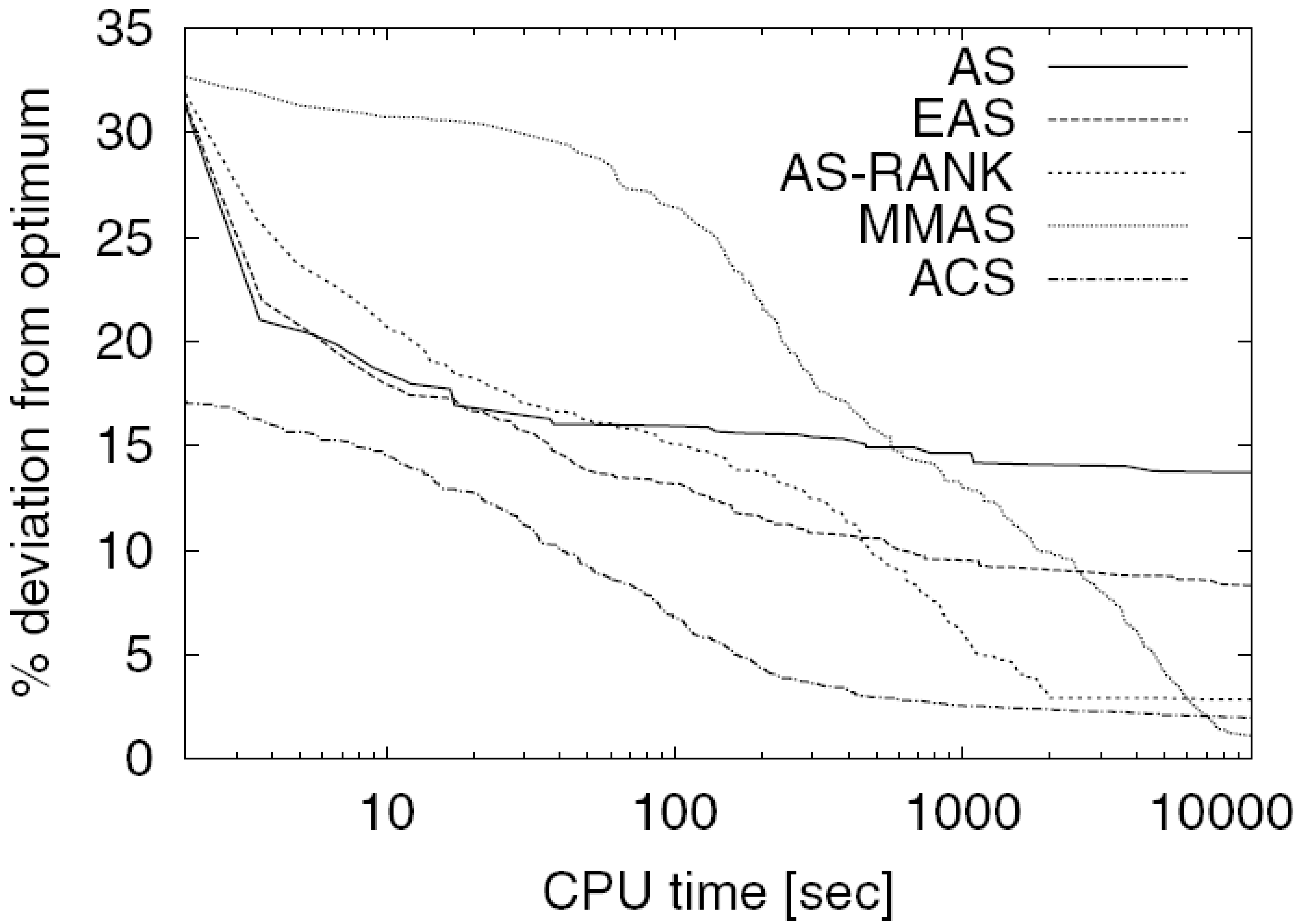
Solution quality of algorithms

- Twenty-five trials for instance **d198**



Solution quality of algorithms

- Five trials for instance **rat783**



Solution quality of algorithms

- We found experimentally that all extensions of AS achieve **much better** final solutions than AS, and in all cases the **worst final solution** returned by the AS extensions is better than the **average final solution** quality returned by AS.
- It can be observed that ACS is the most aggressive of the ACO algorithms and returns the best solution quality for very short computation times.

Behavior of Extensions of AS

- MMAS **initially** produces rather **poor solutions** and in the initial phases it is outperformed even by AS. Nevertheless, its final solution quality, for these two instances, is the best among the compared ACO algorithms.
- Comparisons among the several AS extensions indicate that the best performing variants are MMAS and ACS.

ACO plus Local Search

ACO plus Local Search

- The vast literature on metaheuristics tells us that a promising approach to obtaining high-quality solutions is to couple a **local search algorithm** with a mechanism to generate initial solutions.
- Once ants have completed their solution construction, the solutions can be taken to their local optimum by the application of a local search algorithm.
- Then pheromones are updated on the arcs of the locally optimized solutions.

ACO plus Local Search

- There exist a large number of possible choices when combining local search with ACO algorithms.
- Some of these possibilities relate to the fundamental question of **how effective** and **how efficient** the local search should be.
- In fact, in most local search procedures, the better the solution quality returned, the higher the computation time required.

ACO plus Local Search

- This translates into the question whether for a given computation time
  - it is better to frequently apply a quick local search algorithm that only slightly improves the solution quality of the initial solutions, or
  - whether a slow but more effective local search should be used less frequently.

ACO plus Local Search

- Other issues are related to particular parameter settings and to which solutions the local search should be applied.
- For example, the number of ants to be used, the necessity to use heuristic information or not, and which ants should be allowed to improve their solutions by a local search, are all questions of particular interest when an ACO algorithm is coupled with a local search routine.

ACO plus Local Search

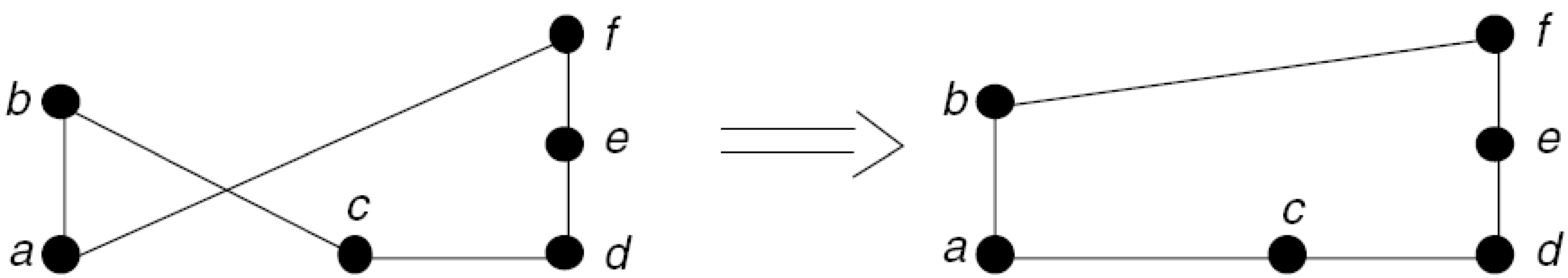
- In general, there may be significant differences regarding particular parameter settings.
- For example, for MMAS it was found that
  - when applied without local search, a good strategy is to frequently use the iteration-best ant to update pheromone trails.
  - Yet, when combined with local search a stronger emphasis of the best-so-far update seemed to improve performance.

ACO plus Local Search

- We study how the performance of MMAS, is improved when coupled with a local search.
- To do so, we implemented three of the most used types of local search for the TSP: **2-opt**, **2.5-opt**, and **3-opt**.
- All three implementations exploit three standard speedup techniques:
  - the use of nearest-neighbor lists of limited length (here 20), the use of a fixed radius nearest-neighbor search, and the use of don't look bits.

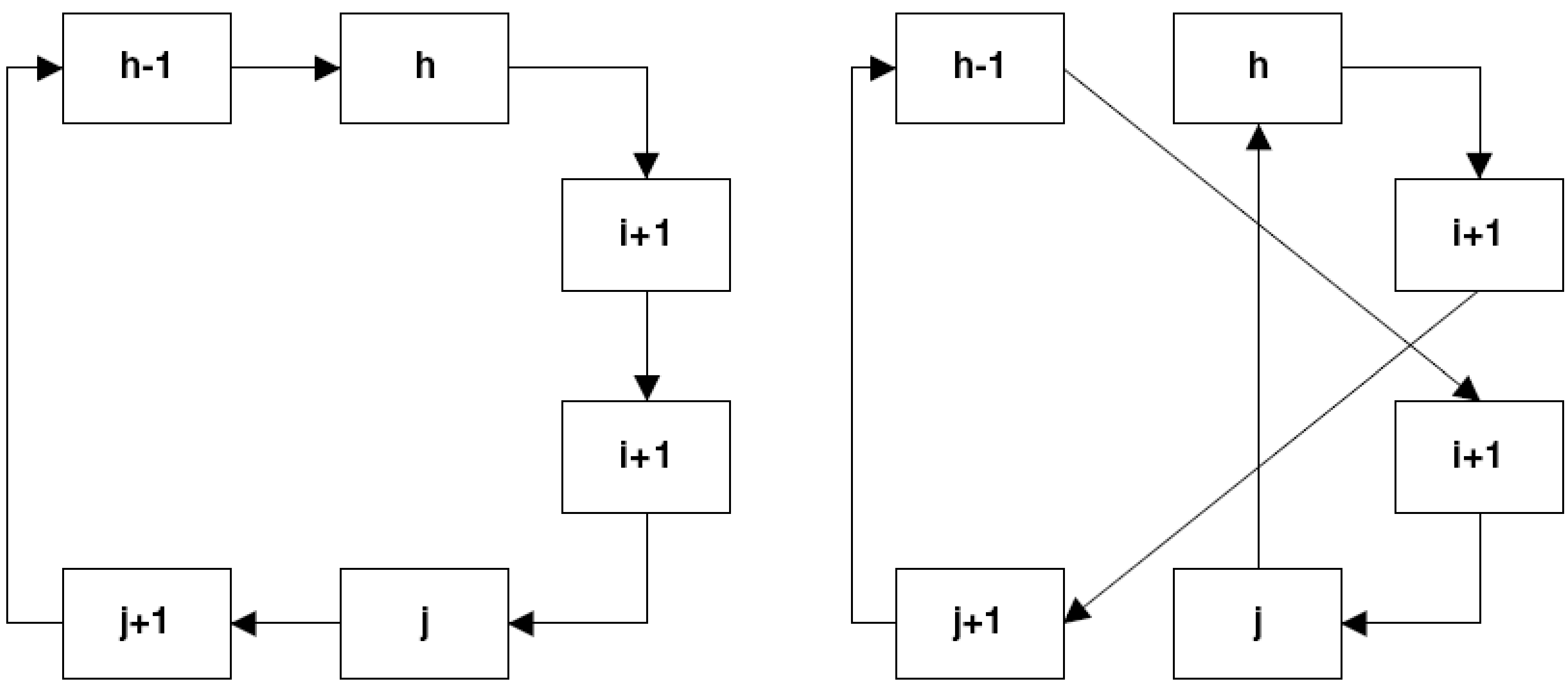
2-opt neighborhood

- The **2–opt neighborhoods** in the TSP
- Given a candidate solution s, the TSP 2–opt neighborhood of a candidate solution s consists of the set of all the candidate solutions s' that can be obtained from s by exchanging two pairs of arcs in all the possible ways.
- Example: the pair of arcs (b, c) and (a, f) is removed and replaced by the pair (a, c) and (b, f)



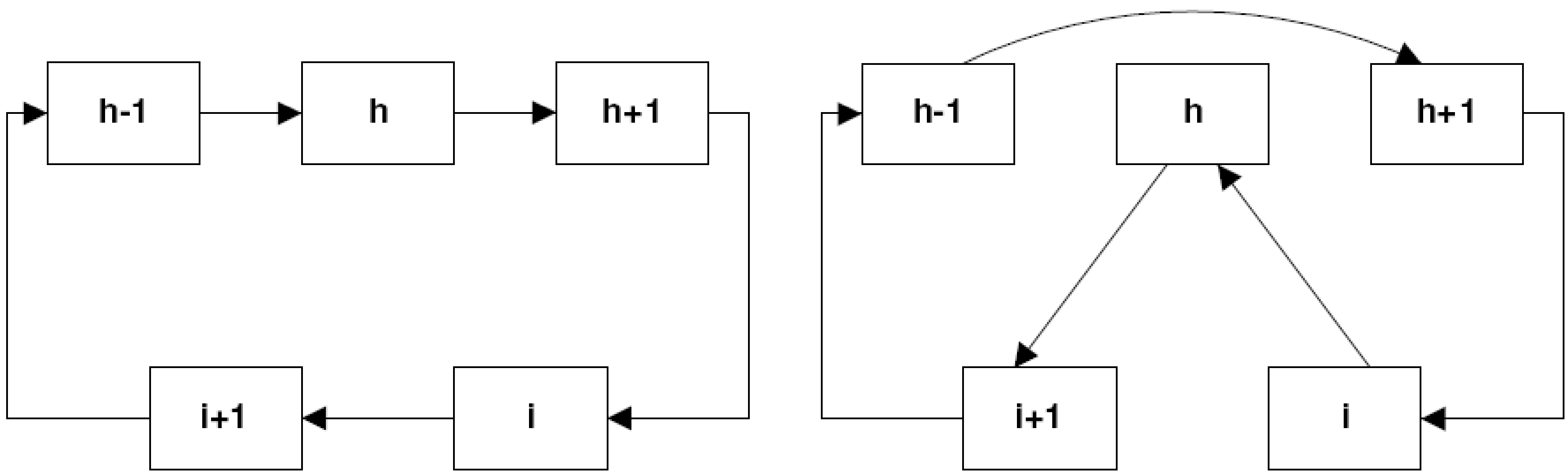
3-opt neighborhood

- The **3-opt neighborhood** consists of those tours that can be obtained from a tour s by replacing at most three of its arcs.
- In a 3-opt local search procedure 2-opt moves are also examined. Example:



2.5-opt neighborhood

- 2.5-opt checks whether inserting the city between a city i and its successor, as illustrated in the figure below, results in an improved tour.



MMAS with 2-opt, 2.5-opt, and 3-opt

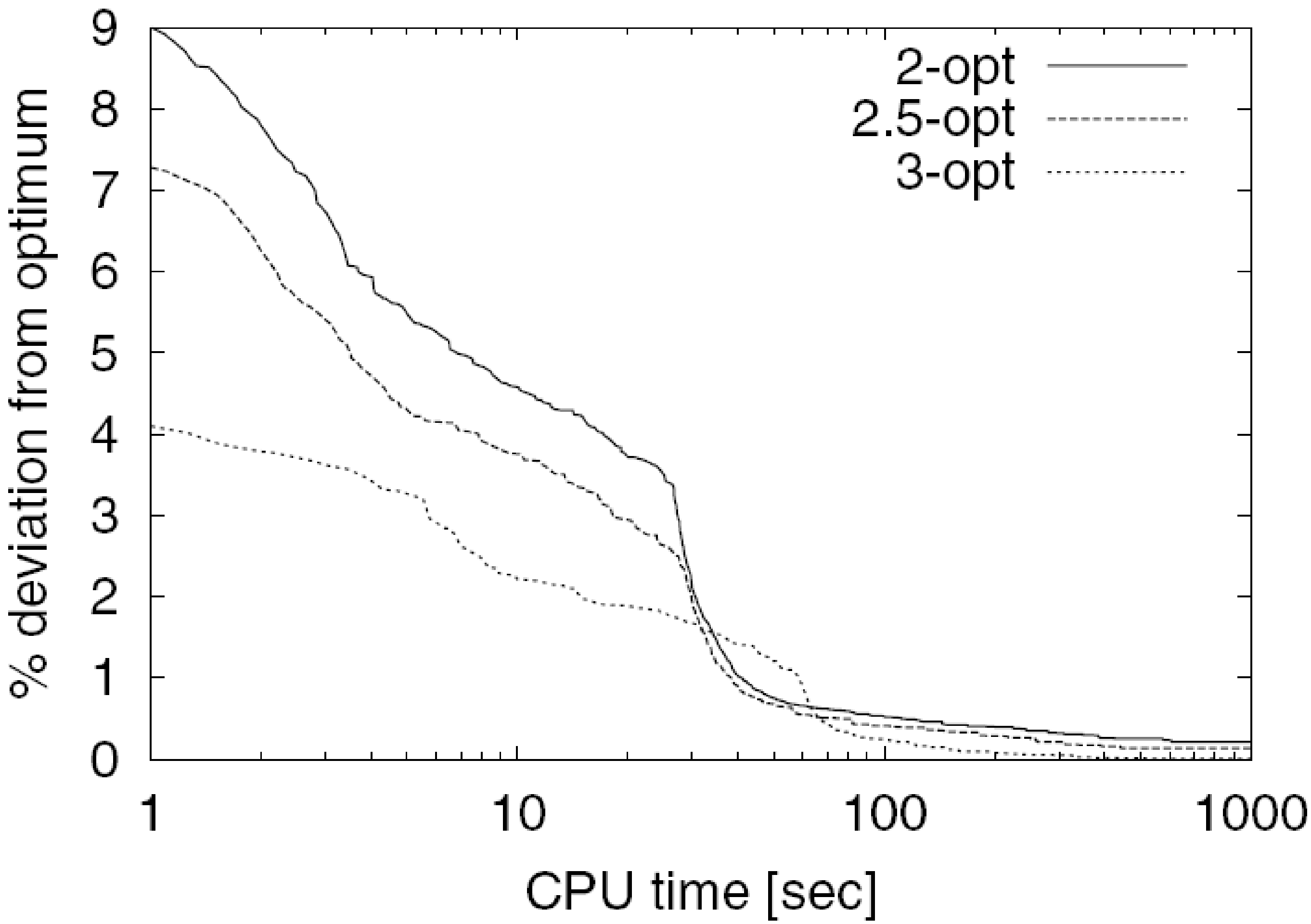
- 2.5-opt leads only to a small, constant overhead in computation time over that required by a 2-opt local search but, as experimental results show, it leads to significantly better tours.
- However, the tour quality returned by 2.5-opt is still significantly worse than that of 3-opt.

MMAS with 2-opt, 2.5-opt, and 3-opt

- We combined MMAS with 2-opt, 2.5-opt, and 3-opt local search procedures.
- While the solution quality returned by these local search algorithms increases from 2-opt to 3-opt, the same is true for the necessary computation time to identify local optima.

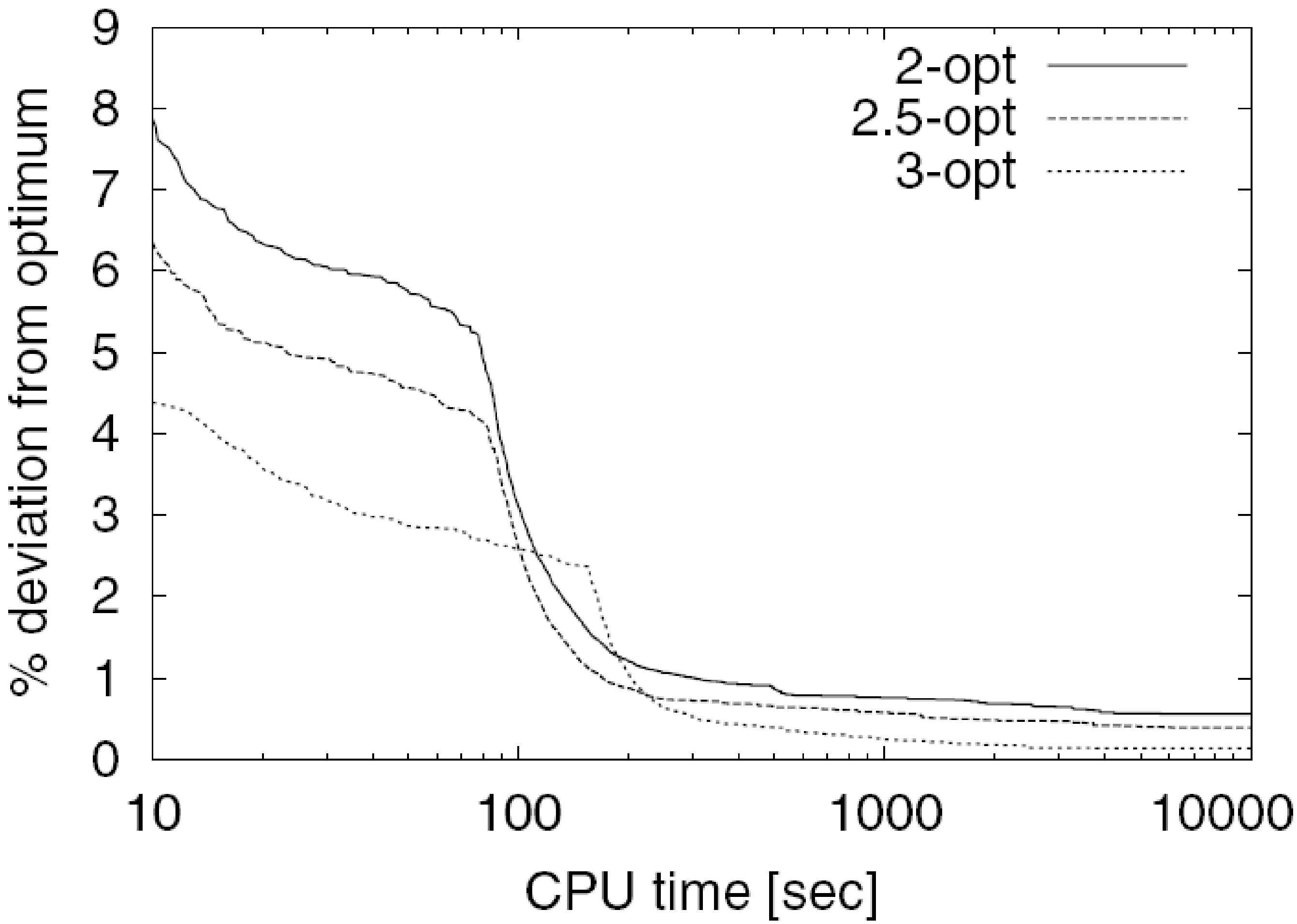
MMAS with 2-opt, 2.5-opt, and 3-opt

- symmetric TSPLIB instances pcb1173



MMAS with 2-opt, 2.5-opt, and 3-opt

- symmetric TSPLIB instances pr2392



MMAS with 2-opt, 2.5-opt, and 3-opt

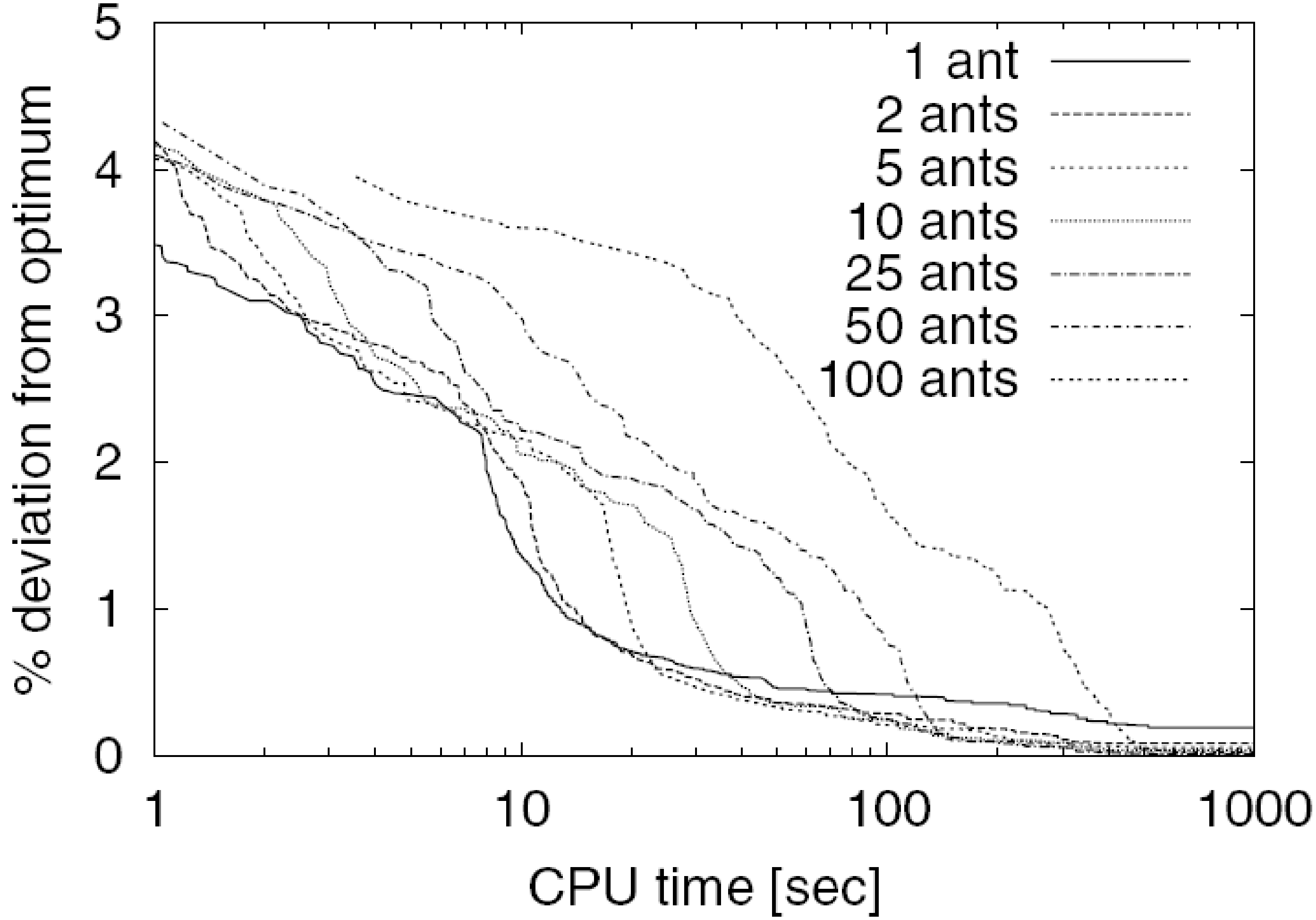
- For the largest amount of computation time, MMAS combined with 3-opt gives the best average solution quality.
- In any case, once the final tour quality obtained by the different variants is taken into account, the computational results clearly suggest that the use of more effective local searches improves the solution quality of MMAS.

Number of Ants

- In a second series of experiments we investigated the role of the number of ants m on the final performance of MMAS.
- We ran MMAS using parameter settings of $m \in \{1, 2, 5, 10, 25, 50, 100\}$ leaving all other choices the same.

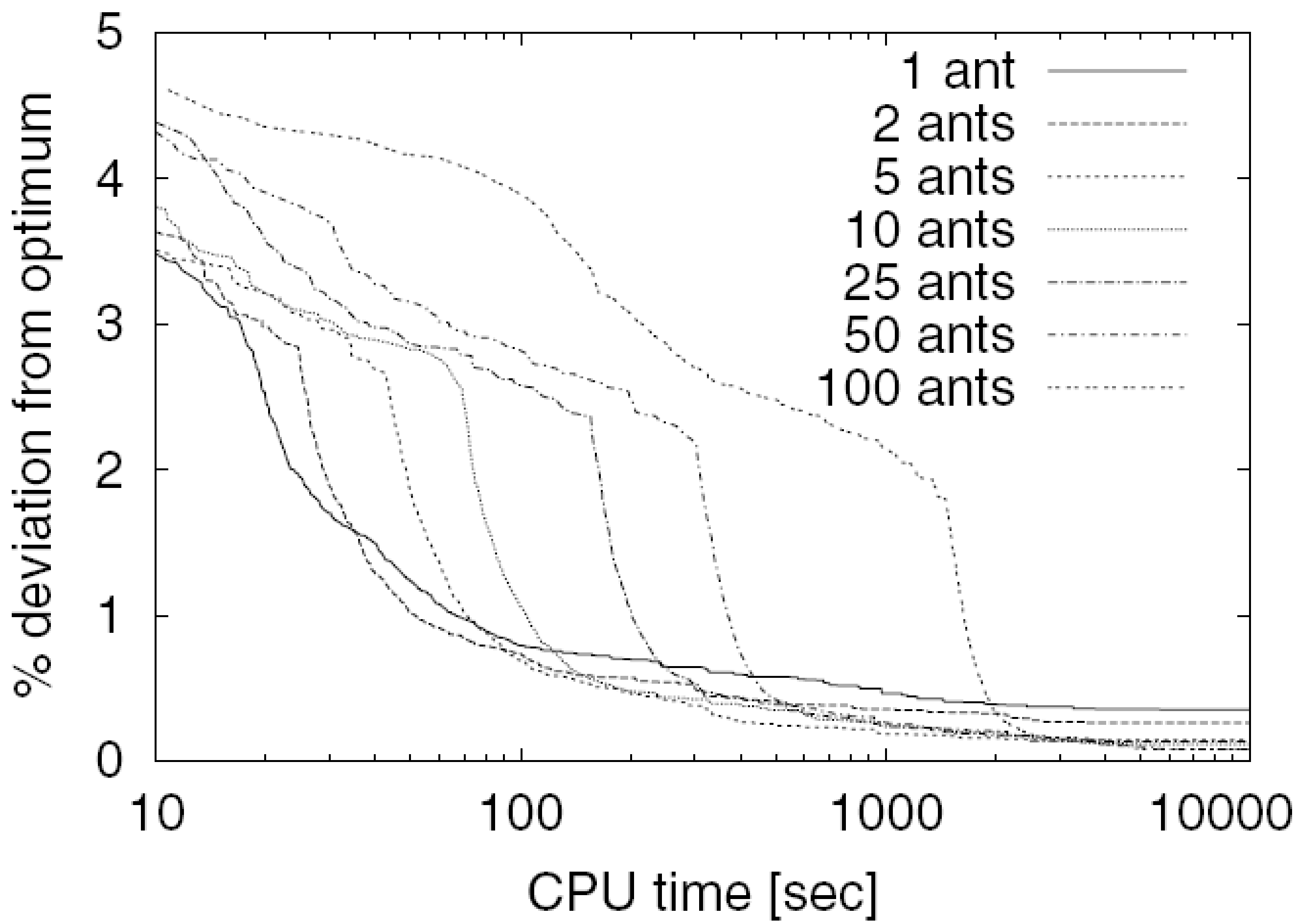
Number of Ants

- symmetric TSPLIB instance pcb1173



Number of Ants

- the symmetric TSPLIB instance pr2392



Number of Ants

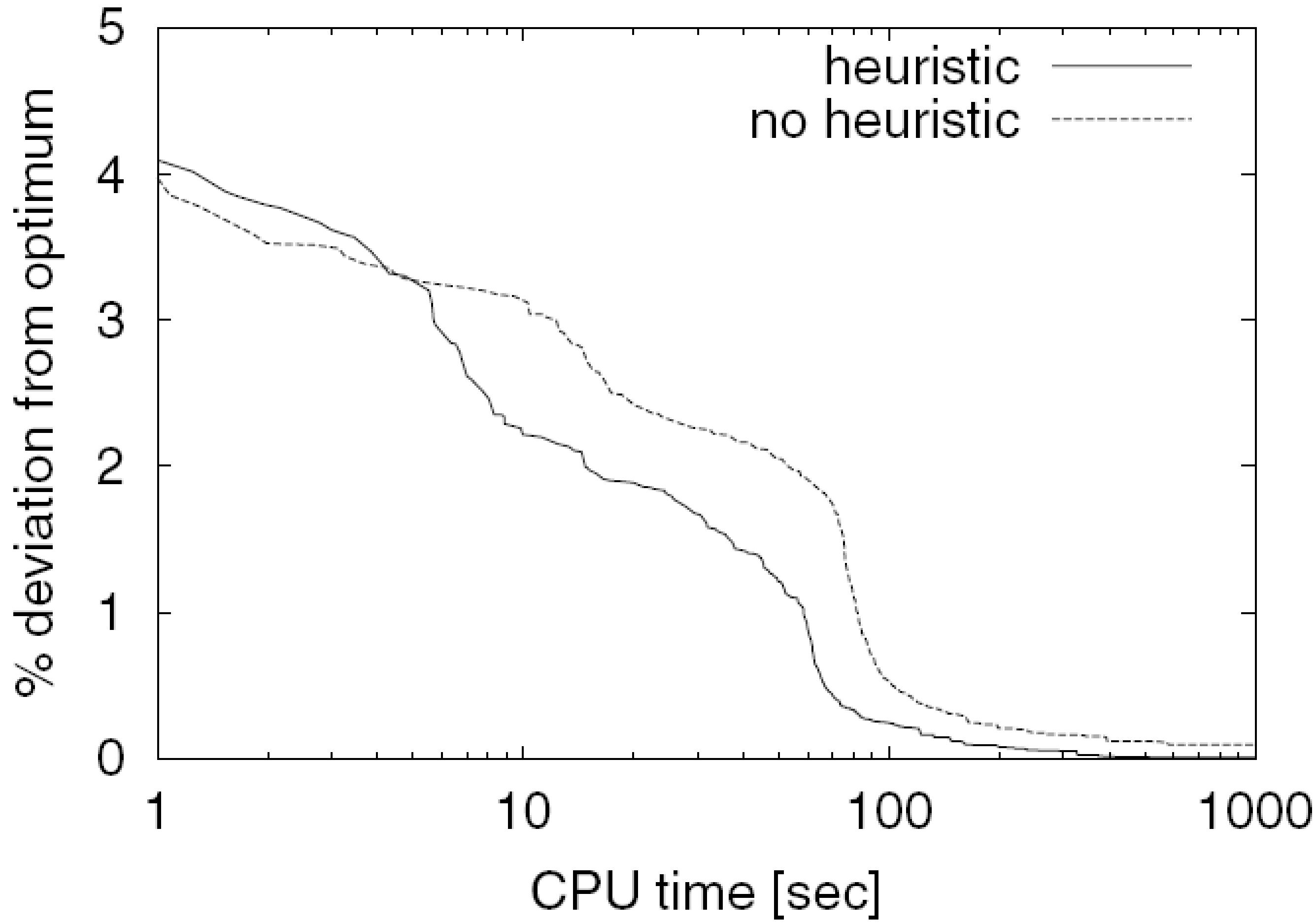
- The result was that on small problem instances with up to 500 cities, the number of ants did not matter very much with respect to the best final performance.
- In fact, the best trade-off between solution quality and computation time seems to be obtained when using a small number of ants—**between two and ten**.

Heuristic Information

- Once local search is added to the ACO implementation, the randomly generated initial tours become good enough.
- It is therefore reasonable to expect that heuristic information is no longer necessary.
- Experiments with MMAS and ACS on the TSP confirmed this conjecture.

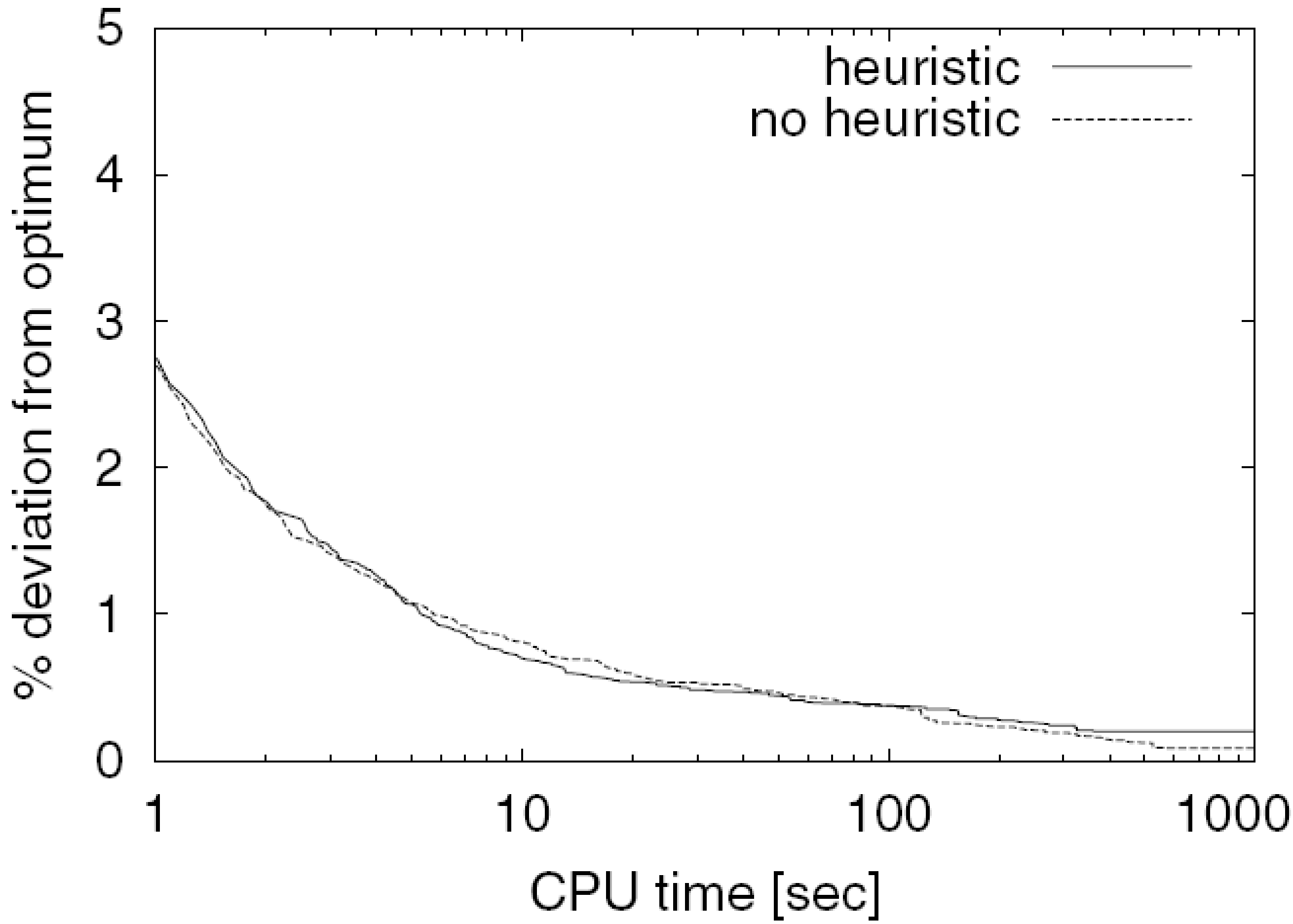
Heuristic Information

- MMAS for the symmetric TSPLIB instance pcb1173



Heuristic Information

- ACS for the symmetric TSPLIB instance pcb1173



References

References

- M. Dorigo and T. Stützle. **Ant Colony Optimization**, MIT Press, Cambridge, 2004.

The End