Ant Colony Optimization

A survey of current research

Dipl. Inf. Gunnar Völkel

Theoretische Informatik
Universität Ulm

10.06.2010
Overview

1. Introduction
   - Foraging behavior of ants
   - Combinatorial Optimization Problem: TSP
   - Metaheuristics

2. Ant Colony Optimization
   - Algorithmic model
   - Description of the metaheuristic
   - Variants of ACO algorithms

3. Theoretical research
   - Convergence
   - Runtime complexity analysis

4. Conclusion
   - Summary
   - Research topics
1 Introduction

- Foraging behavior of ants
- Combinatorial Optimization Problem: TSP
- Metaheuristics
Foraging behavior of ants

Figure: Argentine ant\(^1\)

- in many ant species: single ants able to deposit pheromone perceivable by other ants
- pheromone laying while moving from nest to food source (and vice versa)

⇒ pheromone trails emerge
- guidance for other foragers to food source
- even shortest way to food source optimization

\(^1\) © Joyce Gross
Binary Bridge Experiment

- experiment without food source (Deneubourg et al. [1990])
- pheromone trail establishes on one of the branches

Figure: Percentage of ants per 3-min period passing on the two branches of the bridge (inset). Colony of 1000 workers. (Deneubourg et al. [1990])
Shortcut Experiment

- experiment with food source (Goss et al. [1989])
- advantage gain for the shorter branch due to length difference (ants return earlier)

**Figure:** An ant colony selecting the short branches on both modules of the bridge; (a) one module of the bridge, (b) and (c): photos taken 4 and 8 mins after start. (Goss et al. [1989])
Combinatorial Optimization Problem: Traveling Salesman Problem (TSP)
Combinatorial Optimization Problem: Traveling Salesman Problem (TSP)

TSP \begin{align*}
\left\{ - \left( d(X_k, X_1) + \sum_{i=1}^{k} d(X_i, X_{i+1}) \right) \right\} \rightarrow \max \\
(X_1, \ldots, X_k) \in \Pi(\{v_1, v_2, \ldots, v_k\})
\end{align*}

\( V = \{v_1, \ldots, v_k\} \) set of cities to visit

\( G = (V, E) \) complete graph with cities as vertices (one edge between each city pair)

\( \Pi(\{v_1, \ldots, v_k\}) \) set of permutations of \( V = \{v_1, \ldots, v_k\} \)

\( d : V \times V \rightarrow \mathbb{R}^+ \) distance function (via distance matrix \( D \in (\mathbb{R}^+)^{n \times n} \) possible)

shape of Germany: http://www.mygeo.info/landkarten/deutschland/clipart_deutschland.gif (June 2010)
many important problems $\mathcal{NP}$-complete, e.g.

- Traveling Salesman Problem (TSP)
- Rucksackproblem (KNAPSACK)

$\Rightarrow$ worst-case runtime for exact algorithms: exponential in size of problem instance

approximate algorithms for near optimal solutions at low computational cost

- problem-specific heuristics
- metaheuristics
### Heuristic

A problem-specific method to solve or assist in solving a combinatorial optimization problem (approximately) by:

- Using **knowledge to select** most favorable "search directions" first (search based or brute force algorithms)
- Providing a **criterion for omitting** unpromising "search directions"

### Metaheuristic

A problem-independent algorithmic scheme to solve a combinatorial optimization problem (approximately) by:

- Guidance of the objective function (and usage of heuristics)
- Probabilistic construction of solutions and simultaneous learning about the problem instance
- Probabilistic alteration of solutions while getting feedback from the objective function
<table>
<thead>
<tr>
<th><strong>Heuristic</strong></th>
<th><strong>Metaheuristic</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>a <strong>problem-specific</strong> method <strong>to solve</strong> or <strong>assist in solving</strong> a combinatorial optimization problem (approximately) by</td>
<td></td>
</tr>
<tr>
<td>• using <strong>knowledge to select</strong> most favorable “search directions” first (search based or brute force algorithms)</td>
<td></td>
</tr>
<tr>
<td>• <strong>provide a criterion for omitting</strong> unpromising “search directions”</td>
<td></td>
</tr>
<tr>
<td><strong>Metaheuristic</strong></td>
<td>a <strong>problem-independent</strong> algorithmic scheme <strong>to solve</strong> a combinatorial optimization problem (approximately) by</td>
</tr>
<tr>
<td>• <strong>guidance</strong> of the objective function (and usage of <strong>heuristics</strong>)</td>
<td></td>
</tr>
<tr>
<td>• <strong>probabilistic construction</strong> of solutions and simultaneous <strong>learning</strong> about the problem instance</td>
<td></td>
</tr>
<tr>
<td>• <strong>probabilistic alteration</strong> of solutions while getting feedback from the objective function</td>
<td></td>
</tr>
</tbody>
</table>
Approximate algorithms

in general two main types:

**Construction algorithms**
- start with empty initial solution
- incremental build of solution by adding components until a complete solution is reached
- example: **Greedy Construction Heuristic**
in general two main types:

### Construction algorithms
- start with empty initial solution
- incremental build of solution by adding components until a complete solution is reached
- example: **Greedy Construction Heuristic**

### Local search
- start with complete initial solution
- modifying current best solution to improve it
- current best solution replaced by improved one
- example: **Iterative Improvement**
Approximate algorithms

in general two main types:

Construction algorithms
- start with empty initial solution
- incremental build of solution by adding components until a complete solution is reached
- example: Greedy Construction Heuristic

Local search
- start with complete initial solution
- modifying current best solution to improve it
- current best solution replaced by improved one
- example: Iterative Improvement

- also possible: stopping of exact algorithms before completion
2 Ant Colony Optimization

- Algorithmic model
- Description of the metaheuristic
- Variants of ACO algorithms
Ant Colony Optimization

- metaheuristic inspired by foraging behavior of ants
- developed by M. Dorigo et al. since 1991–1992 (recent survey paper: Dorigo and Stützle [2009])
Ant Colony Optimization

- metaheuristic inspired by foraging behavior of ants
- developed by M. Dorigo et al. since 1991–1992 (recent survey paper: Dorigo and Stützle [2009])

- probabilistic solution construction procedure using
  - (artificial) pheromone trails: dynamically changing at runtime to reflect “search experience”
  - heuristic information about the problem
- one (artificial) ant constructs one solution
Figure: Construction graph
Modelling – Construction Graph: TSP

Figure: Construction graph

Figure: Partial path

Figure: Maximal path

\[ \Phi(\{v_1, v_3, v_6, v_5, v_4, v_2\}) = (1, 3, 6, 5, 4, 2) \]
Modelling – Construction Graph: TSP

Figure: Construction graph

Figure: Partial path

Figure: Maximal path

Figure: Solution decoding
Figure: Construction graph

Figure: Partial path

Figure: Maximal path

Figure: Solution decoding

\[ \Phi(\{v_1, v_3, v_6, v_5, v_4, v_2\}) = (1, 3, 6, 5, 4, 2) \]
Pheromone & Heuristic values

- **pheromone values** $\tau_{ij}$ assigned to each edge $(i, j)$ of the construction graph
- **information on quality** of the single solution component from previous iterations
Pheromone & Heuristic values

**Pheromone values**
- Pheromone values $\tau_{ij}$ assigned to each edge $(i, j)$ of the construction graph
- Information on quality of the single solution component from previous iterations

**Heuristic information values**
- Heuristic information values $\eta_{ij}(u)$ from problem-specific heuristic
- Desirability to choose the single solution component from heuristic point of view
- May depend on partial path $u$
Pheromone & Heuristic values

pheromone values

- **pheromone values** $\tau_{ij}$ assigned to each edge $(i, j)$ of the construction graph
- information on quality of the single solution component from previous iterations

heuristic information values

- **heuristic information values** $\eta_{ij}(u)$ from problem-specific heuristic
- desirability to choose the single solution component from heuristic point of view
- may depend on partial path $u$

- **combination function** $g(\tau, \eta)$ often chosen as

  $$g(\tau, \eta) = \tau^\alpha \cdot \eta^\beta, \quad \alpha, \beta > 0$$

  common choice:

  $$\alpha = 1$$
**Algorithm 1: ACO metaheuristic**

| \( \tau_{ij} = \tau_0 \quad \forall (i, j) \in E \) \quad \text{ // pheromone initialization} |
| \( v_{\text{start}} \in V \) \quad \text{ // start node of } G_c |
| for iteration \( m = 1, 2, \ldots \) do |
| for ant \( s = 1, \ldots, S \) do |
| \( v_{\text{pos}} = v_{\text{start}} \) and \( u = \{ \} \) |
| while feasible continuation \((v_{\text{pos}}, j)\) of \( u \) exists do |
| select \( j \) with probability \( p_{ij} \), where \( i = v_{\text{pos}} \) \quad \text{ // } (i, j) \text{ infeasible} |
| \( p_{ij} = \begin{cases} 
0, & (i, j) \text{ infeasible} \\
\frac{g(\tau_{ij}, \eta_{ij}(u))}{\sum_{(i, r) \in E} g(\tau_{ir}, \eta_{ir}(u))}, & \text{otherwise, } (i, r) \text{ feasible} 
\end{cases} \) |
| \( u = u \oplus (v_{\text{pos}}, j) \) and \( v_{\text{pos}} = j \) |
| end |
| end |
| update the pheromone trails \( \tau_{ij} \) \quad \text{ // variable} |
| end |
**Ant Colony Optimization for TSP**

![Graph](image)

**Figure:** Third step in probabilistic construction procedure

- **current partial path** \( u = \{(1, 3), (3, 6)\} \)
- **probability to choose node** \( i \in \{2, 4, 5\} \) as next:

\[
p_{6,i} = \frac{\tau_{6,i} \cdot \eta_{6,i}(u)^\beta}{\tau_{6,2} \cdot \eta_{6,2}(u)^\beta + \tau_{6,4} \cdot \eta_{6,4}(u)^\beta + \tau_{6,5} \cdot \eta_{6,5}(u)^\beta}
\]

- **possible heuristic:**

\[
\eta_{i,j}(u) = \frac{1}{d(v_i, v_j)}
\]
### Variants of ACO algorithms

#### Variable aspects
- Update set $\Delta S$ selection for pheromone update
- Pheromone update mechanism
  - Usage of local search before update possible
- Per problem (independent of concrete ACO algorithm):
  - Construction graph

#### Two algorithms to be described here
- Ant System (AS)
- $\textit{MAX-MIN}$ Ant System ($\textit{MMAS}$)
Ant System (AS)

- update selection set:
  \[
  \Delta S = \{ \omega^1, \omega^2, \ldots, \omega^S \}
  \]

- pheromone update mechanism:
  \[
  \tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \frac{\rho}{S} \cdot \sum_{s=1}^{S} \Delta \tau_{ij}^s
  \]

  \[
  \Delta \tau_{ij}^s = \begin{cases} 
  C \cdot f(\omega^s) & \text{if } (i, j) \in \omega^s \\
  0 & \text{otherwise}
  \end{cases}
  \]

- explanations:
  \[
  \rho \quad \text{evaporation rate } \rho \in (0, 1)
  \]
  \[
  \omega^s \quad \text{feasible path of ant } s \in \{1, \ldots, S\}
  \]
  \[
  C \quad \text{a constant } C > 0
  \]
**MAX-MIN Ant System (MMAS)**

- **update selection set** ($\mathcal{MMAS}_{bs}$ or $\mathcal{MMAS}_{ib}$):
  \[
  \Delta S = \{ w^{bs} \} \quad \text{or} \quad \Delta S = \{ w^{ib} \}
  \]

- **pheromone update mechanism**:
  \[
  \tau_{ij} = \left[ (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta \tau_{ij}^{best} \right]^{\tau_{max} / \tau_{min}}
  \]

  \[
  \Delta \tau_{ij}^{best} = \begin{cases} 
  C \cdot f(w^{best}) & \text{if } (i, j) \in w^{best} \\
  0 & \text{otherwise}
  \end{cases}
  \]

- **explanations**:
  - $\tau_{min}$, $\tau_{max}$: min./max. pheromone value $\tau_{max} > \tau_{min} > 0$
  - $w^{best}$: best-so-far ($w^{bs}$) or iteration-best ($w^{ib}$)
  - $\rho$: evaporation rate $\rho \in (0, 1)$
  - $C$: a constant $C > 0$
### Application list

<table>
<thead>
<tr>
<th>Problem type</th>
<th>Problem name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing</td>
<td>Traveling salesman (TSP)</td>
</tr>
<tr>
<td></td>
<td>Vehicle routing</td>
</tr>
<tr>
<td></td>
<td>Sequential ordering</td>
</tr>
<tr>
<td>Assignment</td>
<td>Quadratic assignment</td>
</tr>
<tr>
<td></td>
<td>Course timetabling</td>
</tr>
<tr>
<td></td>
<td>Graph coloring</td>
</tr>
<tr>
<td>Scheduling</td>
<td>Project scheduling</td>
</tr>
<tr>
<td></td>
<td>Car sequencing</td>
</tr>
<tr>
<td>Subset</td>
<td>Set covering</td>
</tr>
<tr>
<td></td>
<td>Multiple knapsack</td>
</tr>
<tr>
<td></td>
<td>Maximum clique</td>
</tr>
<tr>
<td>Machine learning</td>
<td>Classification rules</td>
</tr>
<tr>
<td></td>
<td>Bayesian networks</td>
</tr>
<tr>
<td></td>
<td>Neural networks</td>
</tr>
<tr>
<td>Bioinformatics</td>
<td>Protein folding</td>
</tr>
<tr>
<td></td>
<td>DNA Sequencing</td>
</tr>
</tbody>
</table>

**Figure:** Reduced list of applications from Dorigo and Stützle [2009]
Situation until year 2000

- much problem-centered research on ACO algorithms
- several variants of ACO algorithms
- no theory on convergence behavior
- empirical runtime estimations – no theoretical runtime analysis
Theoretical research

- Convergence
- Runtime complexity analysis
Main theoretic questions

1. Does ACO algorithm X converge to an optimal solution?
2. How fast does ACO algorithm X converge? (runtime complexity)
3. What is the influence of the parameters of ACO algorithm X on the convergence speed?
4. If we stop the algorithm before convergence, how good will the solution be? ($\varepsilon$-optimality?)
Definition (Convergence in solution)

\[ P\left( w(\tau(m)) \in \mathcal{W}^* \right) \to 1 \quad (m \to \infty) \]

- evolvement of pheromone matrix \( \tau(m) \)

\[ w(\tau(m)) \ldots \text{constructed path with pheromone matrix } \tau(m) \text{ in iteration } m \]

\( \mathcal{W}^* \ldots \text{set of optimal paths} \)
### Convergence notions

**Definition (Convergence in solution)**

$$P(\ w(\tau(m)) \in \mathcal{W}^* ) \rightarrow 1 \quad (m \rightarrow \infty)$$

- evolvement of pheromone matrix $\tau(m)$

  - $w(\tau(m))$ ... constructed path with pheromone matrix $\tau(m)$ in iteration $m$
  - $\mathcal{W}^*$ ... set of optimal paths

**Definition ($\varepsilon$-Convergence in solution)**

$$\exists M(\varepsilon) \in \mathbb{N}^+ \quad \forall m \geq M(\varepsilon): \quad P(\ w(\tau(m)) \in \mathcal{W}^* ) \geq 1 - \varepsilon \quad (1)$$

- $\varepsilon \in (0, 1)$
Convergence notions

**Definition (Convergence in solution)**

\[ P( w(\tau(m)) \in \mathcal{W}^* ) \rightarrow 1 \quad (m \rightarrow \infty) \]

- evolvement of pheromone matrix \( \tau(m) \)

\( w(\tau(m)) \) … constructed path with pheromone matrix \( \tau(m) \) in iteration \( m \)

\( \mathcal{W}^* \) … set of optimal paths

**Definition (\( \varepsilon \)-Convergence in solution)**

\[ \exists M(\varepsilon) \in \mathbb{N}^+ \quad \forall m \geq M(\varepsilon) : \quad P( w(\tau(m)) \in \mathcal{W}^* ) \geq 1 - \varepsilon \quad (1) \]

- \( \varepsilon \in (0, 1) \)

**Definition (Convergence in value)**

\[ P( \hat{w}(m) \in \mathcal{W}^* ) \rightarrow 1 \quad (m \rightarrow \infty) \]

- some time: optimal solution constructed

\( \hat{w}(m) \) … best-so-far path in iteration \( m \)
Convergence in solution:

- $\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}_{bs,co}$ with $\rho(m) \downarrow$ or $\tau_{\min}(m) \downarrow$. Gutjahr (2002)

$\varepsilon$-Convergence in solution:

- GBAS ($\approx \mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}_{bs}$ with $\tau_{\min} = 0$, $\tau_{\max} = \infty$ and only updates when $\hat{w}$ improves). Gutjahr (2000)

- $\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}_{bs}$ and $\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}_{ib}$ with $\tau_{\min} > 0$. Stützle and Dorigo (2002)

Convergence in value:

- $\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}_{bs}$ and $\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}_{ib}$. Stützle and Dorigo (2002)
Runtime complexity analysis

- idea taken from analysis of *evolutionary algorithms*
- straightforward: application of *Markov chain theory* (according to Gutjahr)
Runtime complexity analysis

- idea taken from analysis of *evolutionary algorithms*
- straightforward: application of *Markov chain theory* (according to Gutjahr)

- usually infeasible: “explosion” of solution space size
- approach: group solutions into classes related to fitness values
possible different fitness values over $S(\mathcal{X})$: $f_1 < f_2 < \cdots < f_d$

$j$th level set: $A_j = \{ s \in S \mid f(s) = f_j \} \quad j \in \{1, \ldots, d\}$

investigation of bound of the staying time in a certain level set $A_j$ only depending on the level number $j$
\( \hat{s} \) best-so-far solution

\( \hat{w} \) corresponding best-so-far path

- assumption \( \hat{s} \in A_j \) \( \Rightarrow \) no change of \( \hat{s} \) until better solution

\( \Rightarrow \) reinforcement only on edges \((a, b) \in \hat{w}\)
\( \hat{s} \) best-so-far solution

\( \hat{w} \) corresponding best-so-far path

- assumption \( \hat{s} \in A_j \implies \) no change of \( \hat{s} \) until better solution

\( \Rightarrow \) reinforcement only on edges \( (a, b) \in \hat{w} \)

- suppose lower bound \( p_j \) for probability of the construction of better solution than \( \hat{s} \) with fitness value \( f_k \) and \( k > j \)
⇒ jump from $A_j$ to $A_k$ with upper bound of expected runtime of

$$\frac{1}{p_j} \quad (= \sum_{r=0}^{\infty} (1 - p_j)^r)$$
⇒ jump from $A_j$ to $A_k$ with upper bound of expected runtime of

$$\frac{1}{p_j} \left( = \sum_{r=0}^{\infty} (1 - p_j)^r \right)$$

- worst-case: $\hat{\psi}$ visits all $A_j$ for $j \in \{1, \ldots, d\}$

⇒ expected worst-case runtime bounded from above by:

$$\frac{1}{p_1} + \frac{1}{p_2} + \ldots + \frac{1}{p_{d-1}}$$

- not applicable for iteration-best strategies, e.g. $\text{MMAS}_{ib}$
Problems with pseudo-Boolean fitness function

Generalized OneMax problem:

\[ f(s) = n + c - d(s, s^*) \rightarrow \max \]

- \( n \in \mathbb{N}^+ \)
- \( s^* \in \{0, 1\}^n \)
- \( d : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1, \ldots, n\} \) as Hamming distance
- \( c \in \mathbb{R}_0^+ \)
Result for Generalized OneMax problem

$\mathcal{MMAS}_{bs,co}$:
- expected runtime in expected number of function evaluations until optimal solution
- parameters:

$$\rho = 1 - \frac{a}{n'}, \quad \tau_{\min} = \frac{a}{n^2}, \quad a > 0 \text{ const.}, \quad \tau_0 = \frac{1}{2n}$$
Result for Generalized OneMax problem

$\mathcal{MMAS}_{bs,co}$:

- expected runtime in expected number of function evaluations until optimal solution

- parameters:

\[
\rho = 1 - \frac{a}{n}, \quad \tau_{\text{min}} = \frac{a}{n^2}, \quad a > 0 \text{ const.}, \quad \tau_0 = \frac{1}{2n}
\]

- expected runtime bounded from above by:

\[
\forall n \geq 2a : \quad \frac{2e^{2a}}{a} \cdot n \cdot H_n = \mathcal{O}(n \log n) \quad \text{with} \quad H_n = \sum_{j=1}^{n} \frac{1}{j}
\]
Conclusion

- Summary
- Research topics
most combinatorial optimization problems hard to solve
Summary

- most combinatorial optimization problems hard to solve

- Ant Colony Optimization (ACO) metaheuristic:
  - repeated probabilistic construction procedure
  - indirect "communication" about solution component quality via pheromone
most combinatorial optimization problems hard to solve

Ant Colony Optimization (ACO) metaheuristic:
- repeated probabilistic construction procedure
- indirect “communication” about solution component quality via pheromone

theoretic results on ACO:
- convergence notions & convergence results for a lot of ACO algorithms
- runtime complexity results for different pseudo-Boolean problems and several algorithms
Research topics I

according to Gutjahr [2008]:

1. extension of runtime complexity results to
   - other (linear and non-linear) pseudo-Boolean functions
   - problems with permutation-structured solutions

2. research on runtime complexity of other ACO variants (ACS, Rank-Based ACO, . . . )

3. influence of parameter choices (evaporation rate \( \rho \), number of ants, . . . )

4. classification of problems based on which metaheuristic performs best
Research topics I according to Gutjahr [2008]:

1. extension of runtime complexity results to
   - other (linear and non-linear) pseudo-Boolean functions
   - problems with permutation-structured solutions

2. research on runtime complexity of other ACO variants (ACS, Rank-Based ACO, ...)

3. influence of parameter choices (evaporation rate \( \rho \), number of ants, ...)

4. classification of problems based on which metaheuristic performs best

- extension of runtime complexity research to \( \mathcal{NP} \)-hard problems
“recent research trends” according to Dorigo and Stützle [2009]:

- Multi-objective optimization
- Dynamic versions of $\mathcal{NP}$-hard problems
- Stochastic optimization
- Continuous optimization
Thank you for your attention.


Part II

Additional slides
1. Detailed definitions
   - Combinatorial Optimization Problem (COP)
   - Construction graph
   - Notation for convergence results

2. Further examples
   - ACO for KNAPSACK
   - Pseudo-Boolean fitness function problems
   - Runtime complexity results for a special TSP instance

3. Comparisons
   - Evolutionary Computation
Detailed definitions

- Combinatorial Optimization Problem (COP)
- Construction graph
- Notation for convergence results
Combinatorial Optimization Problem I

- $X_i \in D_i$ denotes a **decision variable** $X_i$ which can be assigned a value from a **finite** domain $D_i$.

- **decision variable set:**
  $$\mathcal{X} = \{X_1, X_2, \ldots, X_k\}$$

- **domain set:**
  $$\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$$

- **search space:**
  $$S = S(\mathcal{X}) = \bigcup_{v_1 \in D_1} \cdots \bigcup_{v_k \in D_k} \{\{X_1 = v_1, \ldots, X_k = v_k\}\}$$
Combinatorial Optimization Problem I

- $X_i \in D_i$ denotes a **decision variable** $X_i$ which can be assigned a value from a finite domain $D_i$.

- **decision variable set:**

  $$X = \{X_1, X_2, \ldots, X_k\}$$

- **domain set:**

  $$D = \{D_1, D_2, \ldots, D_k\}$$

- **search space:**

  $$S = S(X) = \bigcup_{v_1 \in D_1} \cdots \bigcup_{v_k \in D_k} \{\{X_1 = v_1, \ldots, X_k = v_k\}\}$$

  in terms of **solution components**:

  $$S = S(X) = \bigcup_{c_1 \in C_{X_1}} \cdots \bigcup_{c_k \in C_{X_k}} \{\{c_1, \ldots, c_k\}\}$$
Definition (Combinatorial Optimization Problem)

Given

\[ X = \{X_1, X_2, \ldots, X_k\}, \quad D = \{D_1, D_2, \ldots, D_k\} \quad \text{and} \quad S = S(X) \]

then

\[
(P) \begin{cases} 
  f(s) \to \max \\
  s = \{c_{X_1}, \ldots, c_{X_k}\} \in S(X), \quad X_i \in D_i \quad \forall i \in \{1, \ldots, k\} \\
  \omega(c_{X_1}, \ldots, c_{X_k}) = \text{true}, \quad \forall \omega \in \Omega
\end{cases}
\]

with

\[ f : S \to \mathbb{R}_0^+ \quad \text{objective function} \]

\[ \Omega \quad \text{finite set of constraints among the variables} \]

\[ S_\Omega \subseteq S \quad \text{set of feasible solutions:} \]

\[ S_\Omega = \{s \in S \mid \omega(s) = \text{true} \quad \forall \omega \in \Omega\} \]

is called a \textbf{combinatorial optimization problem (COP)}. 44 / 56
Definition: Construction graph

- Algorithmic model description taken from Gutjahr [2007]

Definition (Construction graph, solution decoding function)

Graph $G_c = (V, E)$ together with solution decoding function $\Phi$ is called construction graph if and only if the following holds:

1. $G_c$ has unique start node.
2. $\mathcal{W}$ – set of directed, feasible paths $w$ in $G_c$ satisfying:
   1. $w$ starts at the start node of $G_c$,
   2. $w$ contains each node of $G_c$ at most once,
   3. $w$ is maximal in $\mathcal{W}$.

$$\Phi : \mathcal{W} \rightarrow S_{\Omega} \text{ surjective}$$

$\Rightarrow$ to each feasible solution $s \in S_{\Omega}$ corresponds at least one feasible path $w \in \mathcal{W}$ such that $\Phi(w) = s$

Partial path a path $u$ in $G_c$ that is not maximal

Feasible continuation of $u$ for $u$ with last node $i$, an edge $(i, j) \in E$ with $\exists w \in \mathcal{W} : w = u \oplus (i, j)$
\[ \tau(m) = (\tau_{ij}(m))_{(i,j) \in E} \]  
vector of pheromone trails in iteration \( m \)

\[ \hat{\omega}(m) \]  
best-so-far path in iteration \( m \)

\[ \omega(\tau) \]  
random path of an ant with given pheromone trails \( \tau = (\tau_{ij})_{(i,j) \in E} \)

\[ \mathcal{W}^* \]  
set of optimal paths w.r.t. objective function \( f \)

\[ f^* \]  
optimal objective function (or fitness) value

\[ P(A) \]  
probability of an event \( A \)

- **feasible path and its corresponding solution used synonymously**
Further examples

- ACO for KNAPSACK
- Pseudo-Boolean fitness function problems
- Runtime complexity results for a special TSP instance
KNAPSACK: Problem definition

\[ \begin{align*} 
\sum_{i=1}^{k} X_i \cdot u_i & \rightarrow \text{max} \\
(X_1, \ldots, X_k) & \in \{0, 1\}^k \\
\sum_{i=1}^{k} X_i \cdot w_i & \leq W 
\end{align*} \]

Objects \(1, 2, \ldots, k \in \mathbb{N}^+\)

\(X_i\) decision whether object \(i \in \{1, \ldots, k\}\) is included \((D_i = \{0, 1\})\)

\(u_i \in \mathbb{R}^+\) value for usefulness of object \(i \in \{1, \ldots, k\}\)

\(w_i \in \mathbb{N}^+\) weight of object \(i \in \{1, \ldots, k\}\)

\(W \in \mathbb{N}^+\) capacity of knapsack

Figure: Construction graph for KNAPSACK (chain construction graph)

$\nu_i$ “pre-decision node” for item $i$

$\nu_i^1$ “post-decision node” meaning item $i$ is included

$\nu_i^0$ “post-decision node” meaning item $i$ is not included

This graph: independent decisions whether to include an item
**Figure:** Construction Graph for KNAPSACK with associated pheromone values

- edges without pheromone values: automatic transition

- probabilities:

\[
p_{v_i, v_i^b} = \frac{g(\tau_{ib}, \eta_{ib}(u))}{g(\tau_{i1}, \eta_{i1}(u)) + g(\tau_{i0}, \eta_{i0}(u))}
\]

\[
p_{v_i^b, v_{i+1}} = 1
\]
Needle in a Haystack (NH) problem: Let $x^* \in \{0, 1\}^n$ fixed.

$$f(x) = I\{x = x^*\} \rightarrow \max, \quad x \in \{0, 1\}^n$$

$k$-Needles in a Haystack (k-NH) problem: Let $x^*_1, \ldots, x^*_k \in \{0, 1\}^n$ be $k$ different fixed solutions.

$$f(x) = I\{x = x^*_1 \vee \ldots \vee x = x^*_k\} \rightarrow \max, \quad x \in \{0, 1\}^n$$

Generalized OneMax problem: Let $x^* \in \{0, 1\}^n$ fixed, $d$ as Hamming distance and $c \in \mathbb{R}_0^+$.

$$f(x) = n + c - d(x, x^*) \rightarrow \max, \quad x \in \{0, 1\}^n$$

NH-OneMax problem: Let integers $n$ and $k \leq n$ be given.

$$f(x) = \left( \prod_{i=1}^{k} x_i \right) \cdot \left( \sum_{i=k+1}^{n} x_i + 1 \right) \rightarrow \max, \quad x \in \{0, 1\}^n$$

LeadingOnes problem:

$$f(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \quad x \in \{0, 1\}^n$$
Runtime complexity results for a special TSP instance

\[ D = \begin{pmatrix}
\infty & 1 & n & n & \cdots & n & n \\
n & \infty & 1 & n & \cdots & n & n \\
n & n & \infty & 1 & \cdots & n & n \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
n & n & n & n & n & \infty & 1 \\
n & n & n & n & n & n & \infty \\
1 & n & n & n & n & n & \infty \\
\end{pmatrix} \]

**Figure**: distance matrix of special TSP instance

- Published in Zhou [2009] – ACO algorithm: \( \approx M M A S_{bs} \)

- Without heuristic, expected runtime complexity:
  \[ \mathcal{O}(n^6 + \frac{1}{\rho} \cdot n \cdot \ln n) \]

- With heuristic \( \eta_{ij}(u) = \frac{1}{d_{ij}} \), expected runtime complexity:
  \[ \mathcal{O}(n^5 + \frac{1}{\rho} \cdot n \cdot \ln n) \]
Comparisons

- Evolutionary Computation
Comparison with Evolutionary Computation

similarities:
- population of individuals
- knowledge about the problem collected by the population

differences:
- EC: all knowledge in population
- ACO: memory of past performance maintained in pheromone trails
- “Local search” (Mutation) and Crossover (EC) vs Construction algorithm (ACO)
encoding of solutions, common approaches:

- binary strings
- arrays of integers or decimal numbers

**Algorithm 2: (1+1) EA for pseudo-Boolean function problems**

\[
p = \frac{1}{n} \quad x \in \mathbb{R} \{0, 1\}^n
\]

```plaintext
while stopping criterion not met do
  for \( i \in \{1, \ldots, n\} \) do
    \[ x'_i = \begin{cases}
      \neg x_i, & \text{with probability } p \\
      x_i, & \text{with probability } (1 - p)
    \end{cases} \]
    if \( f(x') > f(x) \) then \( x = x' \)
  end
end
```