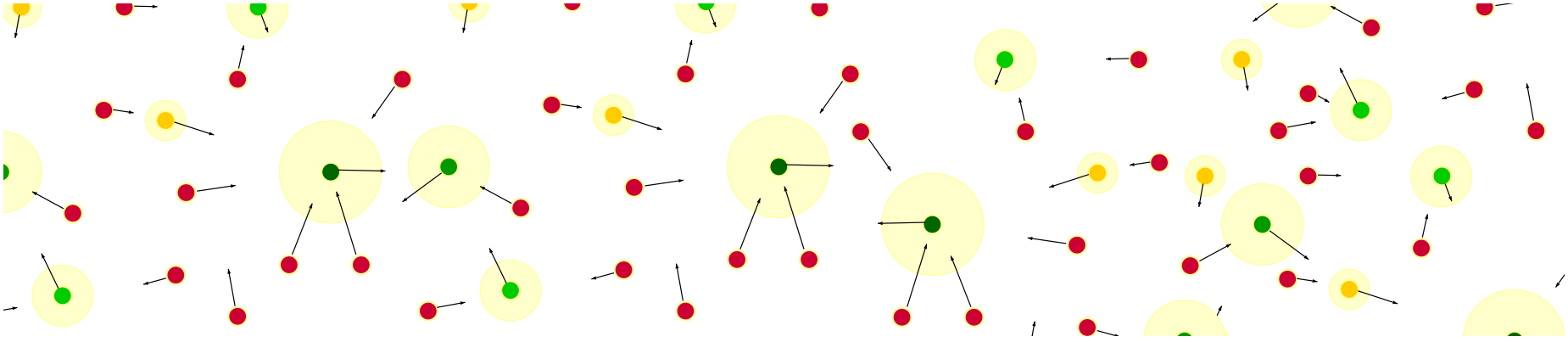
Freie Universität Berlin



Metaheuristiken (Gruppe 3-1)

Verfahren zur näherungsweisen Lösung von Optimierungsproblemen

David Bohn, Luca Keidel, Fabian Reimeier

Institut für Informatik

27. Februar 2015

Was ist eine (Meta-)heuristik?

- Heuristik:
  - Suche von guten (d.h. nahezu optimalen) Lösungen für ein Optimierungsproblem in kurzer Zeit
  - Aber: *keine* Garantie für Gültigkeit oder Optimalität
  - Oftmals problembezogen
- Metaheuristik:
  - Problemunabhängiges Optimierungsverfahren
    - $\Rightarrow$ Problembezogene Teile werden gekapselt
  - Problemunabhängiger Algorithmus steuert problembezogenen Algorithmus
  - Unterliegen aber ebenfalls den selben Beschränkungen wie Heuristiken

Problemstellung

- Generische Implementierung verschiedener Metaheuristiken
  - physikalisch motiviert:
    - *Simulated Annealing*
  - von Prozessen in der Natur motivierte Metaheuristiken:
    - evolutionsbasiert: *Genetische Algorithmen*
    - schwarmbasiert: *Particle Swarm Optimization, Firefly Algorithmus*
- Test der implementierten Metaheuristiken anhand einfacher Probleme:
  - Test anhand des Traveling Salesman Problems
- Verwendung im Kontext von Stapeln und Packen:
  - Stapeln beliebiger Polygone unter Rotation
  - Packen beliebiger Polygone unter Translation und Rotation

Simulated Annealing

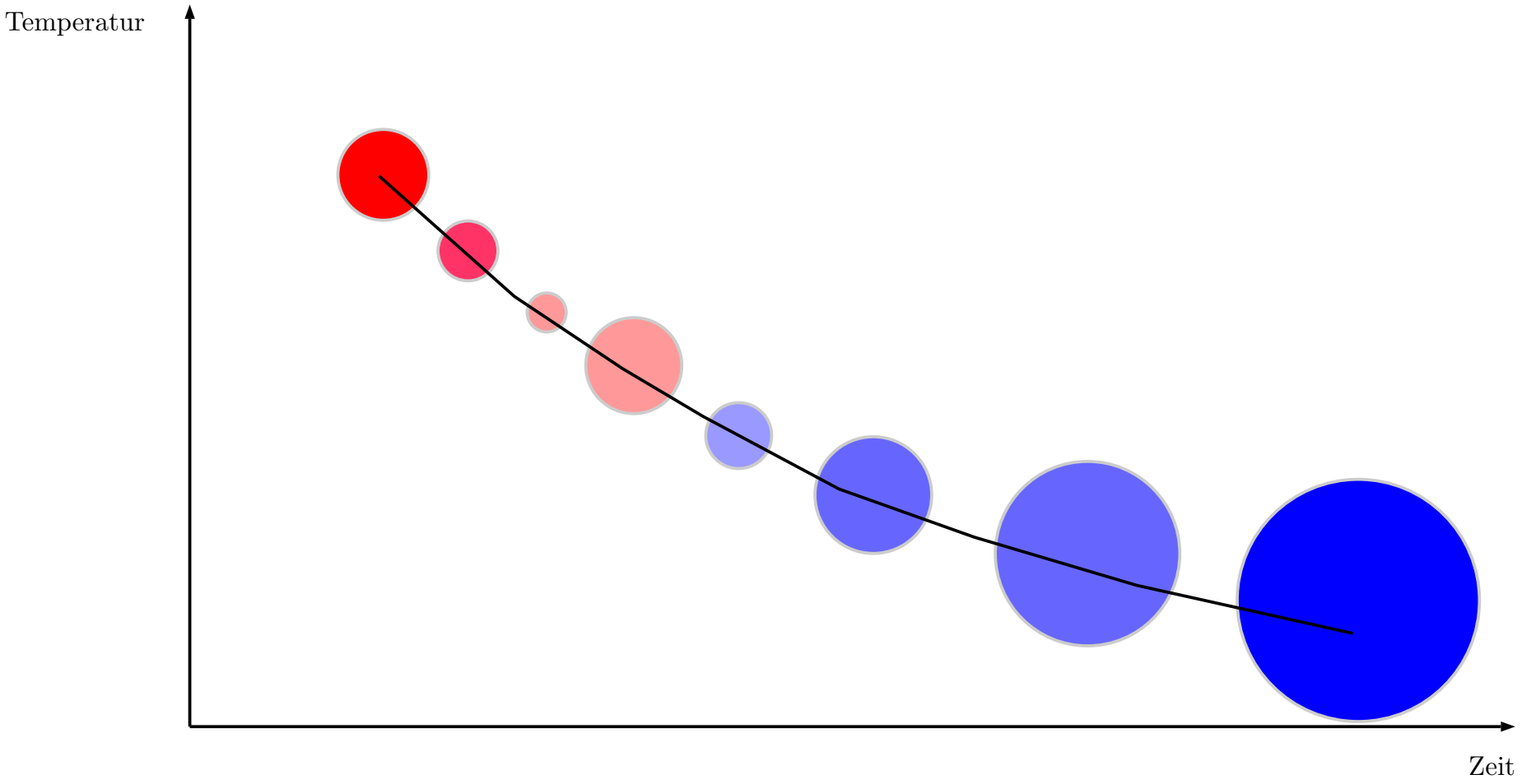


Abbildung: Idee von Simulated Annealing

Simulated Annealing

- Motiviert durch phys. Vorgänge in Metallverarbeitung
- Metall wird zunächst stark erhitzt, dann langsam abgekühlt
  - $\Rightarrow$ Atome bilden stabile Kristalle
- Grober Ablauf:
  1. Als Startlösung wird eine zufällige Lösung gewählt
  2. Solange das Abbruchkriterium nicht erfüllt ist:
     - 2.1 Generiere neue Lösung (unter Einbeziehung des Zufalls)
     - 2.2 Akzeptiere neue Lösung wenn sie besser ist oder per Zufallsentscheidung
- Im Verlauf des Algorithmus können auch schlechtere Lösungen akzeptiert werden
  - Wahrscheinlichkeit eine schlechtere Lösung zu akzeptieren sinkt je weiter die Temperatur sinkt

Simulated Annealing - Abkühlungskurven

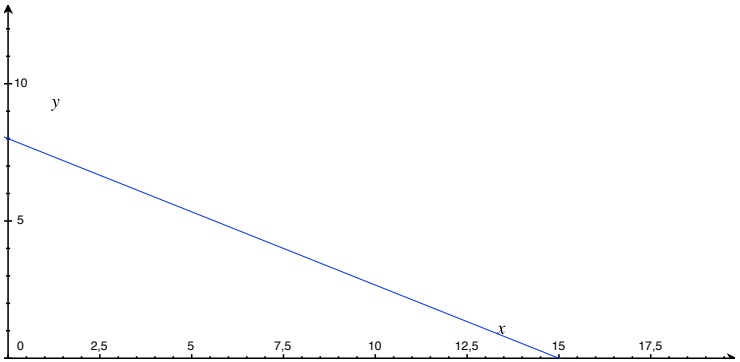


Abbildung: Lineare Abkühlung $T(t) = T(0) \cdot \left(1 - \frac{t}{N}\right)$

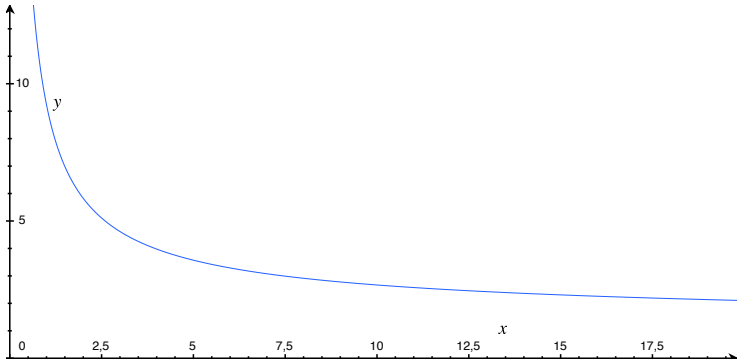


Abbildung: Logarithmische Abkühlung $T(t) = T(0) \cdot \frac{\alpha}{\ln(1+t)}$

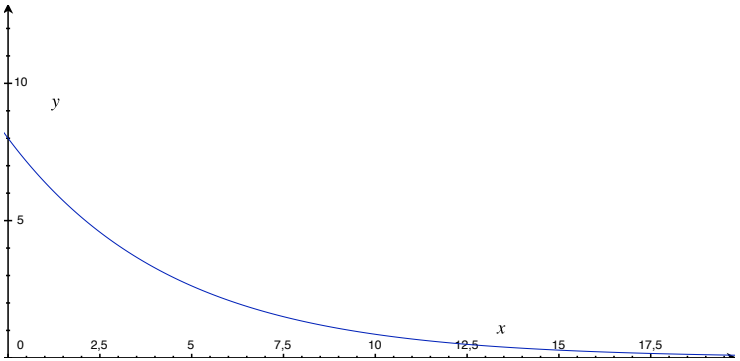


Abbildung: Geometrische Abkühlung $T(t) = T(0) \cdot \alpha^t$

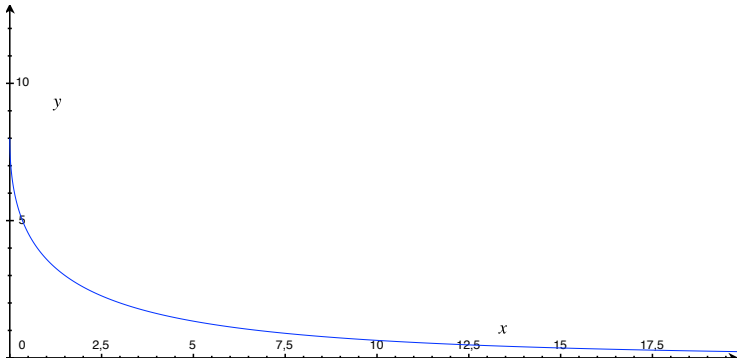


Abbildung: Exponentielle Abkühlung $T(t) = T(0) \cdot e^{-\alpha\sqrt{t}}$

Genetische Algorithmen

- Optimierungsverfahren basierend auf genetischer Evolution
- Beginnt mit einer Population (=Menge) von möglichen Lösungen
- Zustände der Population entwickeln sich auf zwei Arten:

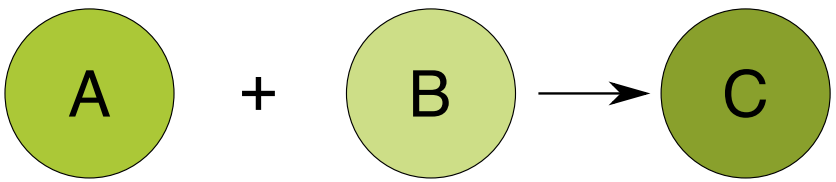


Abbildung: Kreuzung

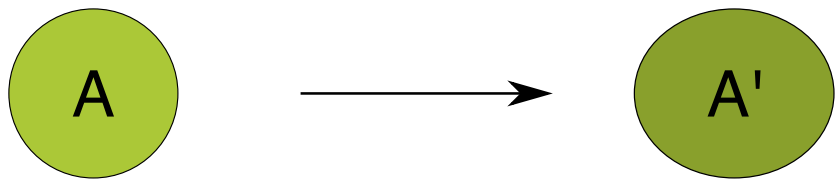


Abbildung: Mutation

- Die neuen Zustände verdrängen u.U. existierende Zustände (survival of the fittest)
- Population degeneriert (d.h. es gibt keine Veränderungen mehr)
  - Lösung: Insular GA
    - Verteilung der Population auf einzelne Inseln
    - Degenerieren die Inseln, werden Zustände ausgetauscht

Particle Swarm Optimization

- Basiert auf Verhaltungsbeobachtung von Individuen in Vogelschwärmen

  $\Rightarrow$ Evaluation, Vergleich und Imitation

- Individuen werden im Algorithmus als Partikel dargestellt, jeder Partikel repräsentiert eine mögliche Lösung
- Partikel können sich mit individueller Geschwindigkeit bewegen in bel. Richtung, können sich ihre jemals beste erreichte Position merken
- Partikel haben drei Möglichkeiten zur Bewegung:

  1. Lokale Suche
  2. Orientierung am eigenen Optimum
  3. Orientierung am globalen Optimum

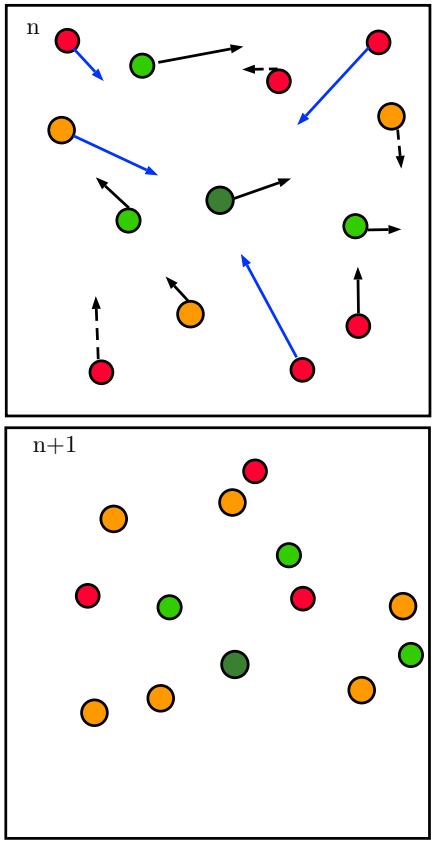


Abbildung: Zwei aufeinanderfolgende Zustände des Suchraums

Firefly Algorithm

- Glühwürmchen:
  - Fliegen herum und leuchten
  - Fühlen sich von Licht angezogen und fliegen auf dieses zu
  - Je heller die Lichtquelle, desto größer die Anziehung
- Glühwürmchen Algorithmus:
  - Ein Glühwürmchen ist Lösung
  - Die Leuchtkraft entspricht der Güte der Lösung
  - Die Bewegung eines Glühwürmchens ist die Veränderung der Lösung
    - Entweder zufällig oder zu einem helleren Glühwürmchen
    - Durch die Bewegung verändert sich die Helligkeit des Glühwürmchens
  - Je heller das Glühwürmchen scheint, desto weiter reicht sein Licht

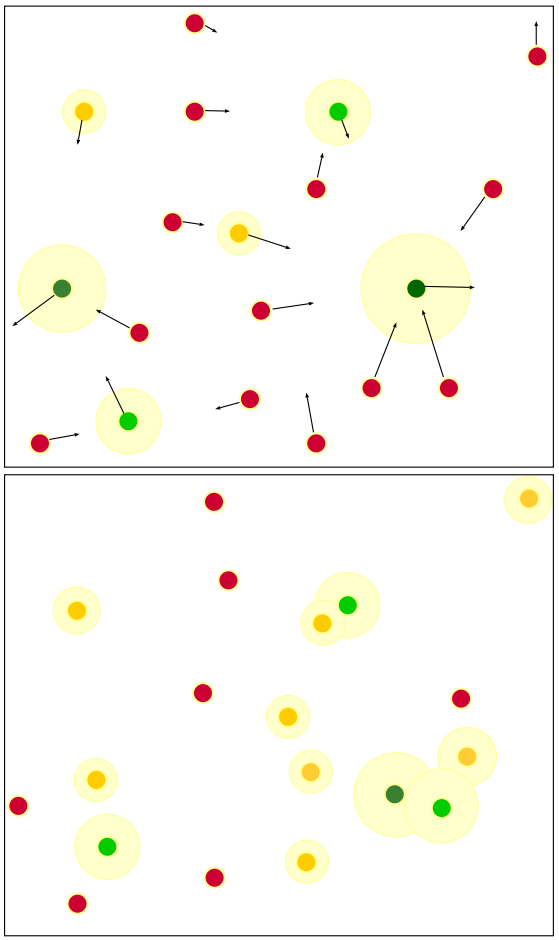


Abbildung: Bewegung der Glühwürmchen

Nicht umgesetzte Algorithmen

- Ameisenalgorithmus
- Inspiriert durch das Verhalten von Ameisenkolonien bzw. Ameisenstraßen:
  - Ameisen einigen sich auf kürzesten gefundenen Weg mithilfe von Pheromonen
  - Übertragen auf Algorithmus:
    - Ameisen suchen den kürzesten Weg im Entscheidungsgraphen
- Problem:
  - Algorithmus setzt diskreten Suchraum voraus

  $\Rightarrow$ Für Stapeln und Packen ungeeignet (kontinuierlicher Suchraum)

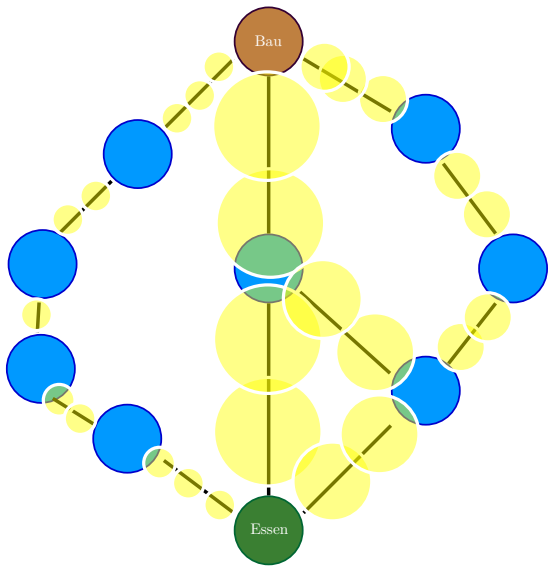


Abbildung: Beispielhafte Pheromonspur

Nicht umgesetzte Algorithmen

- Scatter Search:
  - arbeitet auf zufällig generierten Referenzgruppen
  - verbessert zuerst vorhandene Lösungen der Referenzgruppen
  - findet dann neue Lösungen durch lineare Kombination von Lösungen aus zuvor gebildeten Untermengen der Referenzgruppen
- Problem:

  $\Rightarrow$ Aufwand für die Umsetzung ist im Vergleich zum relativ geringen Nutzen für das Projekt zu hoch

Generische Controller

- Controller
  - steuern die Ausführung einer Metaheuristik
  - können in den Algorithmus eingreifen und den Algorithmus manipulieren
    - Verändern der Lösung möglich
    - Einfügen von neuen Lösungen zur Erhöhung der Varianz
- Implementierte Controller:
  - Fixed Length Controller (alle Metaheuristiken)
    - Ausführung für eine feste Anzahl von Schritten
  - Stuck Terminate Controller (Simulated Annealing und PSO)
    - Ausführung solange bis für n Schritte keine bessere Lösung mehr gefunden wird
  - Target Fitness Controller (Simulated Annealing (experimentell))
    - Ausführung solange, bis die Lösung eine gewisse Güte erreicht hat

Traveling Salesman Problem

- Problem des Handlungsreisenden:
  - Handlungsreisender will n Städte jeweils genau einmal besuchen
  - Befindet sich dabei in einer Startstadt, zu der er abschließend zurückkehrt
  - Gesucht ist die kürzeste Route zwischen den n Städten
- Mathematische Modellierung als gewichteter, ungerichteter Graph
  - Knoten: Städte
  - Kanten: Verbindung zwischen den Städten, Gewicht entspricht Distanz
- TSP ist NP-schweres Optimierungsproblem:
  - Zielfunktion: Summe der Kantengewichte der Route

GUI zur Visualisierung des TSP

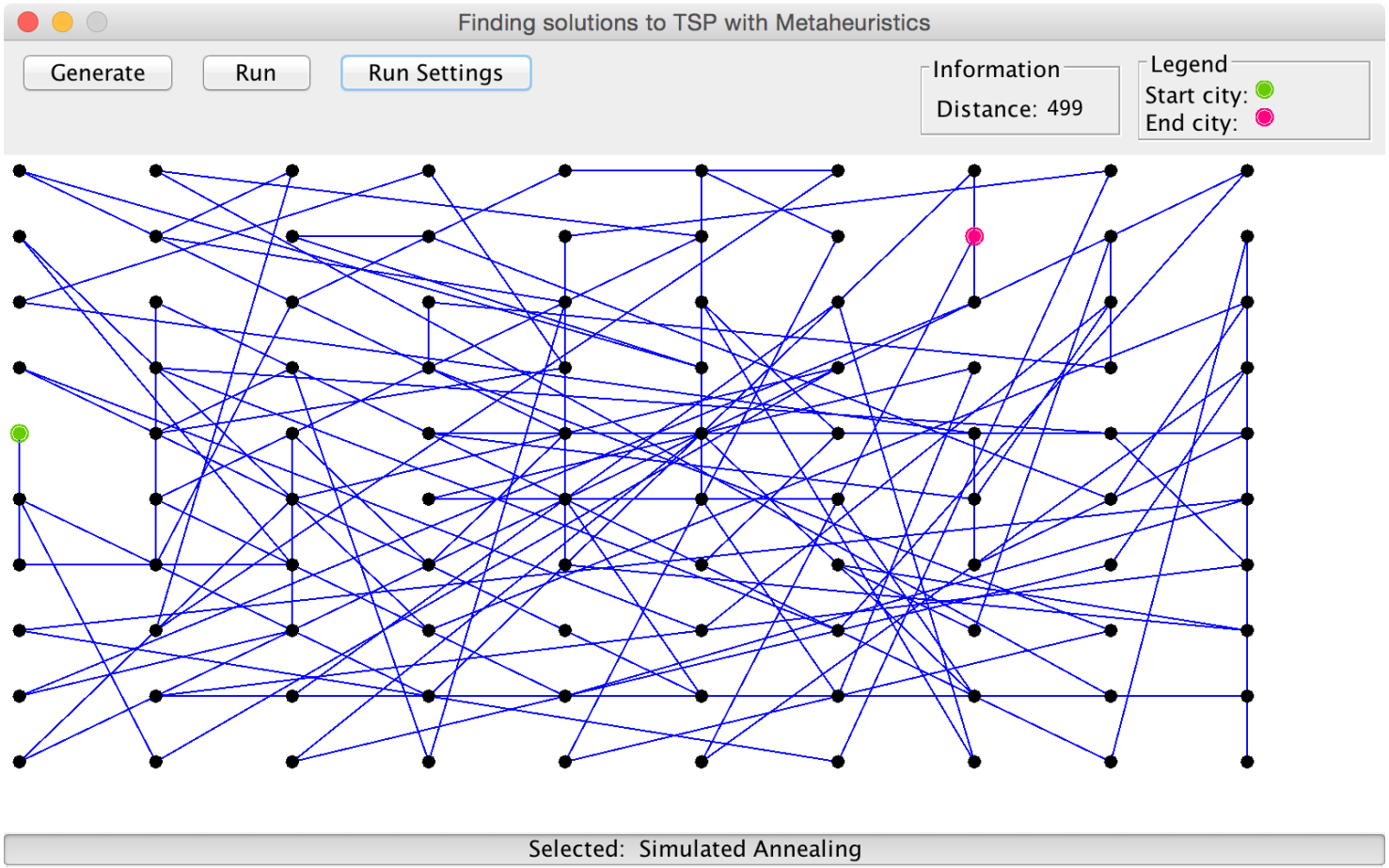


Abbildung: Visualisierung des TSP, Knoten sind in horizontaler und vertikaler Richtung jeweils 1 Einheit entfernt $\Rightarrow$Kürzeste Route 100

Stapeln beliebiger Polygone

- Verwendung von allen implementierten Metaheuristiken
- Zielfunktionen:
  - Konvexe Hülle
  - Bounding Box
  - Achsenparallele Bounding Box
- Ablauf:
  - Initiales Stapeln der Polygone auf ihrem Schwerpunkt im Mittelpunkt
    - bei nichtkonvexen Polygone wird der Schwerpunkt der kovexen Hülle verwendet
  - Anwendung einer Metaheuristik unter Verwendung von Mutator-Klassen

Stapel-Mutator

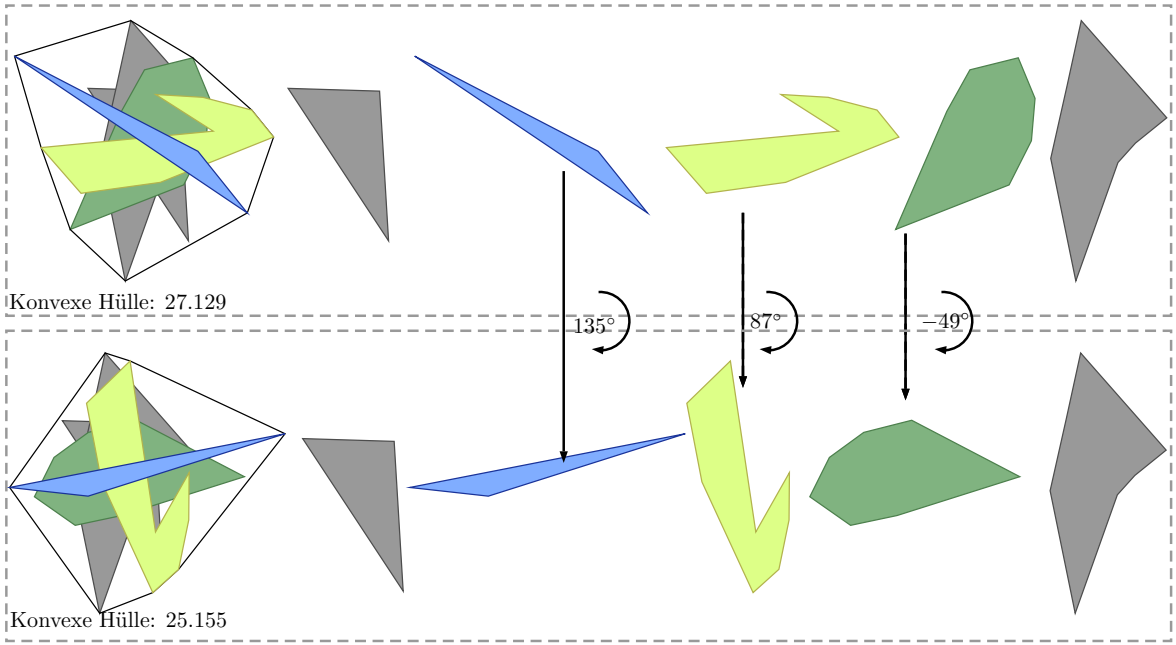


Abbildung: Mutation durch Rotation von drei Polygonen

- Einfacher Mutator: Überführung eines Zustands in einen neuen Zustand
  - Auswahl einer zufällig großen Teilfolge der Polygone
  - Rotiere jedes Polygon um einen zufälligen Winkel $\alpha \in [-180°, 180°]$

Kreuzung

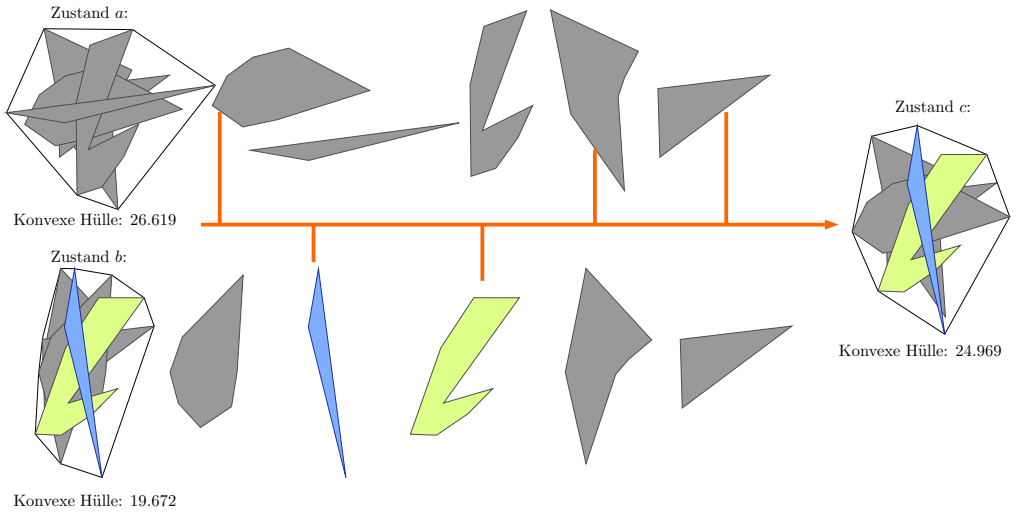


Abbildung: Kreuzung von a und b

- Kreuzung: Verschmelze zwei Zustände a und b zu einem neuen Zustand c
  - hier: Annäherung einer schlechteren a an eine bessere Lösung b
  - Auswahl einer zufällig großen Teilmenge S aus der Menge der Polygone P mit $|P| = n$ und $|S| \in \left[1, \frac{n}{2}\right]$
  - Für jedes in Polygon $s \in S$: Ersetze äquivalentes Polygon in a durch s

  $\Rightarrow$ Annäherung durch Kopie eines Teils der besseren Lösung

Packen beliebiger Polygone

- Verwendung von allen implementierten Metaheuristiken
- Zielfunktionen:
  - Konvexe Hülle
  - Bounding Box
  - Achsenparallele Bounding Box
- Ablauf:
  - Initiales Stapeln der Polygone auf ihrem Schwerpunkt im Mittelpunkt
  - Anwendung einer Metaheuristik unter Verwendung von Mutator-Klassen
    - Anordnung der Polygone können durch Translation oder Translation und Rotation verändert werden
    - Überlappungen werden in Lösung zugelassen, aber durch Zielfunktion bestraft

Pack-Mutator

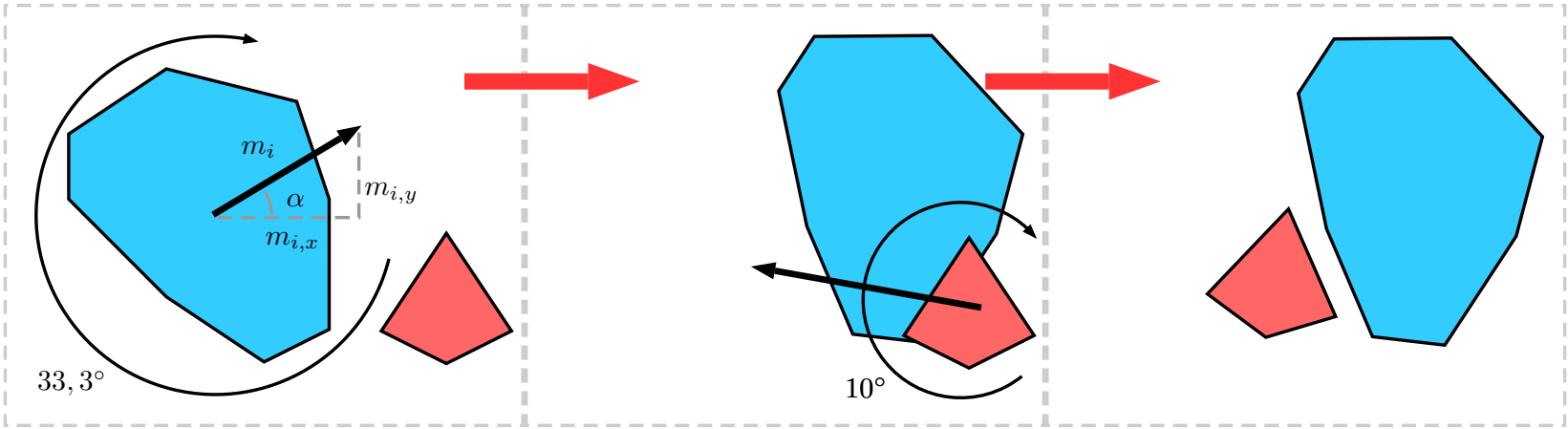


Abbildung: Mutation durch Rotation und Translation

- Einfacher Mutator: Überführung eines Zustands in einen neuen Zustand
- Position jedes Polygons i: $\vec{p_i} = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$
- Mutieren der Position jedes Polygons durch Vektoraddition:
  - $\vec{p_i'} = \vec{p_i} + \vec{m_i}$, wobei $\vec{m_i} = \begin{pmatrix} |\vec{m_i}| \cdot \cos\alpha \\ |\vec{m_i}| \cdot \sin\alpha \end{pmatrix}$ mit $|\vec{m_i}| \in [1, 100]$ und $\alpha \in [0°, 360°]$ zufällig sind

Kreuzung

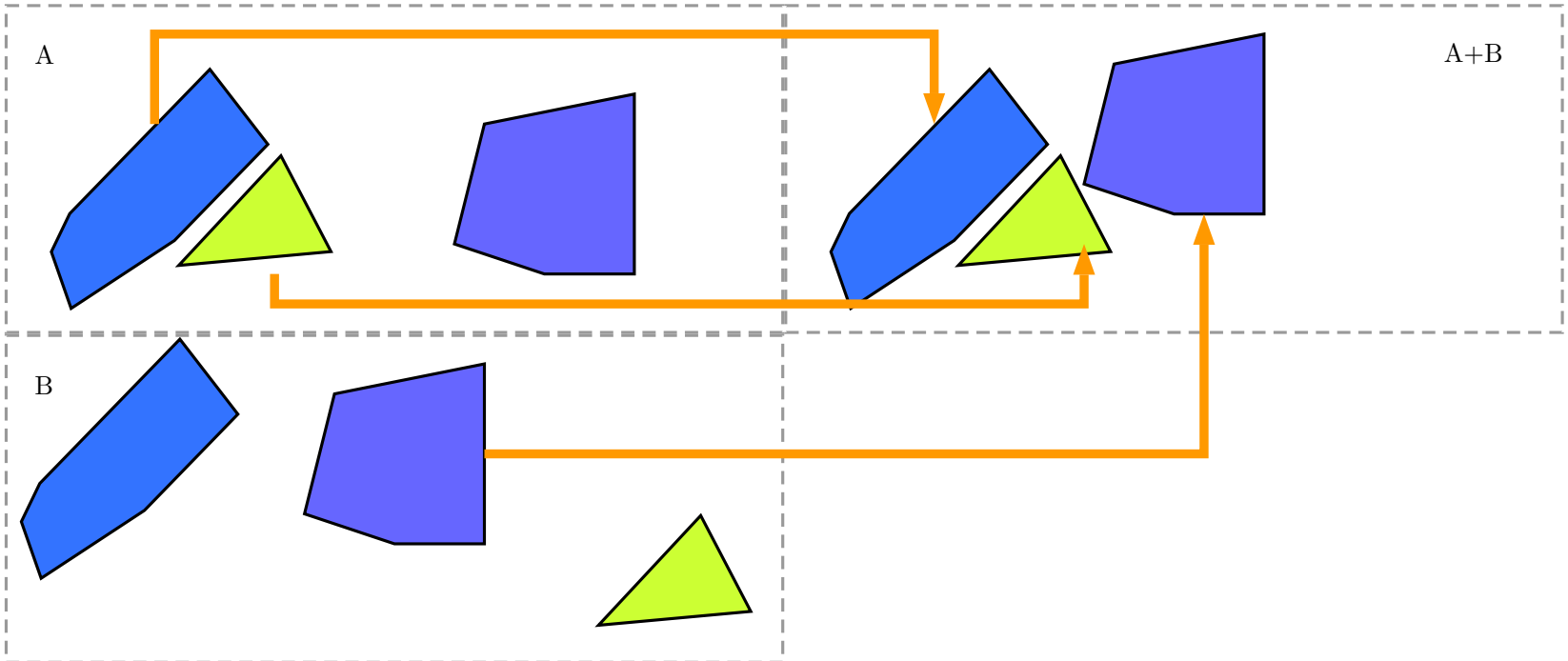


Abbildung: Kreuzung zweier Zustände

- Kreuzung: Verschmelze zwei Zustände a und b zu einem neuen Zustand c
- Verfahren analog zur Kreuzung zweier Zustände beim Stapeln

  $\Rightarrow$ Annäherung durch Kopie eines Teils der besseren Lösung

Zielfunktionen

- Optimierung der konvexen Hülle oder Bounding Box $\Rightarrow$ Zielfunktion
- Bestrafung von Überlappungen
  - Überlappung $\Rightarrow$ ungültige Lösung
- Strategien:
  - Addition der größten Fläche auf bisherigen Wert der Zielfunktion für jede Überlappung
  - Addition der konvexen Hülle auf bisherigen Wert der Zielfunktion für jede Überlappung
  - Bestrafung abhängig von der Überlappungsfläche

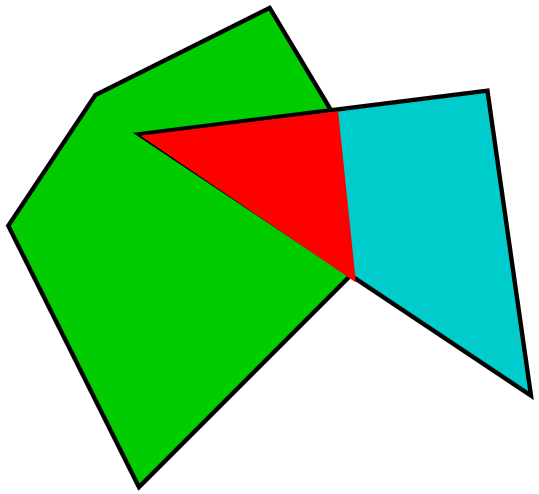


Abbildung: Zu bestrafende Überlappung

One more thing...

Polygonerkennung in Pixelgrafiken

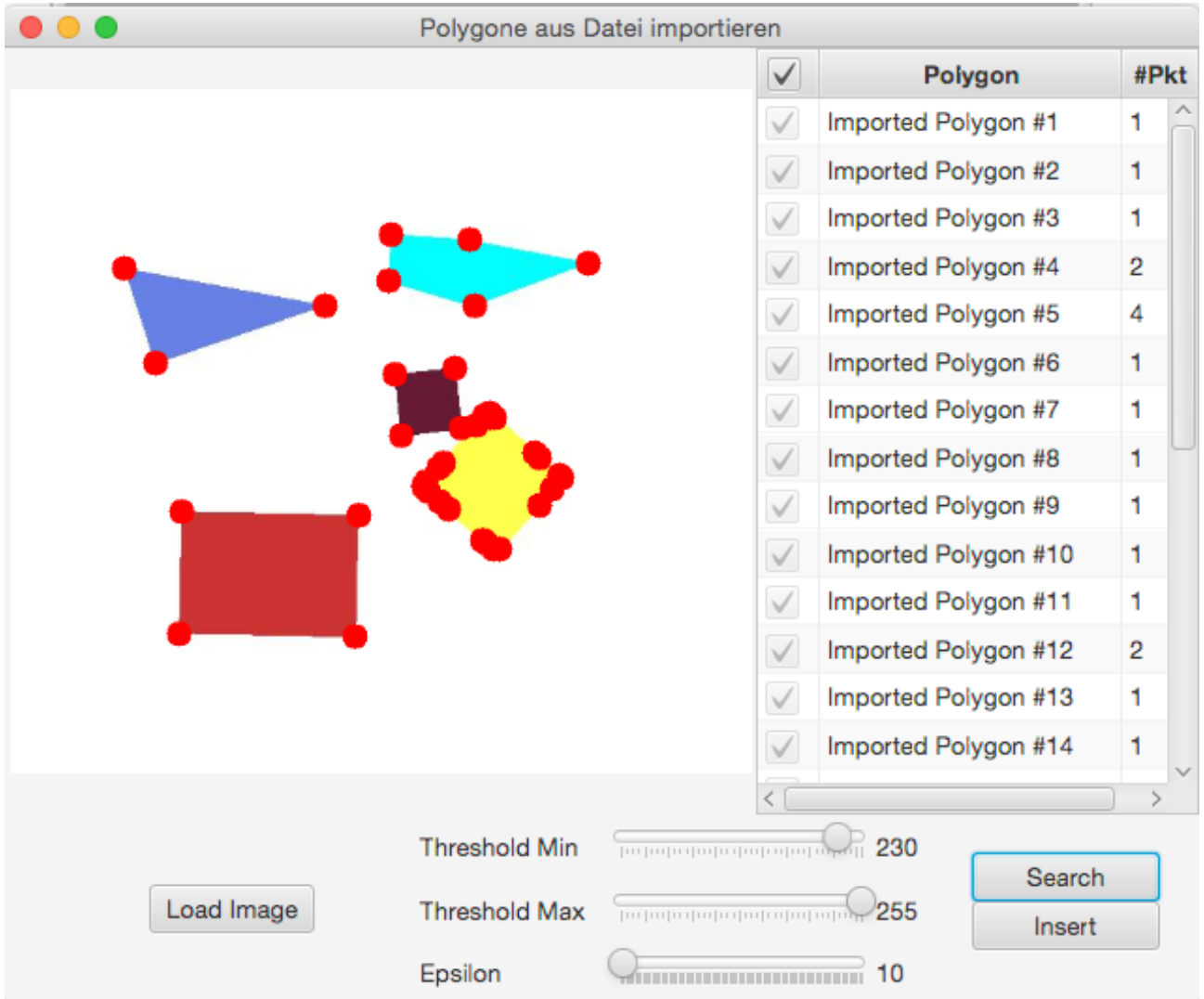


Abbildung: Dialog zur Polygonerkennung

Polygonerkennung in Pixelgrafiken

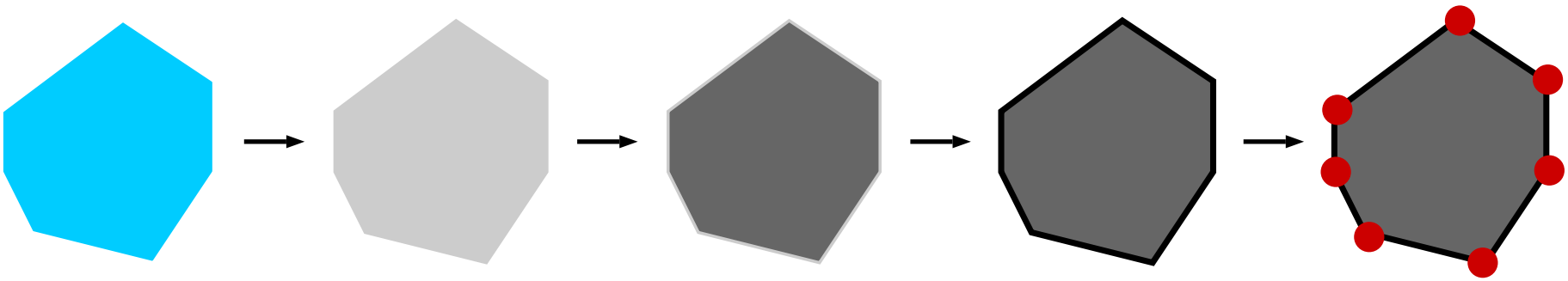


Abbildung: Schritte der Erkennung

1. Umwandlung des Bildes in Graustufen
2. Schwellenwert-Korrektur
3. Finden der Konturen
4. Polynom-Approximation aus den Punkten mit dem Douglas-Peucker-Algorithmus

- Zur Arbeit mit Pixelgrafiken wurde die Computer Vision-Bibliothek OpenCV verwendet.

Export als Grafik

- ermöglicht den Export einer Polygongruppe als Vektorgrafik (SVG) oder Rastergrafik (PNG und JPEG)
- Verfahren zum Export von Vektorgrafiken:
  1. Bestimmung der Maße der Polygongruppe
  2. Erzeugung einer neuen XML-Datei mit SVG Tags
  3. Für jedes Polygon wird ein SVG-Polygon generiert:

```
<poly points="..." fill="..." id="..." stroke="..." fill-opacity="" />
```

- Verfahren zum Export von Rastergrafiken:
  1. Bestimmung der Maße der Polygongruppe
  2. Anlegen eines neuen `BufferedImage`
  3. Zeichnen jedes Polygons in das `BufferedImage`
  4. Export über native Java-Funktionen in gewünschtes Format

Fragen?