Genetische Algorithmen

- Verwandte Verfahren
  - Ameisenkolonie-Optimierung
    - indirekte Kommunikation durch Modifikation der Umgebung
    - Pfäromon
    - Kürzester Pfad in Graph
    - Verdunstung möglich
    - Verbesserungen
      - Elistismus
        - Zusätzliches Pfäromon für derzeit beste Rundreise
      - Rangbasiert
        - Nach Güte der Lösung sortieren
        - Nur beste Ameisen dürfen Markieren
        - Schränkt Suchreim ein!
          - Obere Grenze für Pfäromon
      - Lokale Suche
  - Partikelschwarm-Ptimierung
    - n-dimensionaler Raum
    - Basiert auf sozialer Interaktion
    - Kandidaten sind Partikel
    - Position + Geschwindigkeit
    - Nachbarschaften
      - Ring
        - Jeweils 2 Nadchbarn
      - Stern
        - Jeder mit jedem
    - Neue Geschwindigkeit pro Iteration berechnen
      - Basiert auf:
        - Viskosität
        - Aktuelle Geschwindigkeit
        - Zufall * pbest(i)
        - Zufall * pbest(global)
        - Neue Position berechnen
- Schematheorem
  - Schema
    - Ordnung o(S)
      - Anzahl von 0en und 1er
    - Distanz d(S)
      - Abstand zwischen erster/letzter 0/1
  - Fitness
    - Avg aller Instanzen
    - Nur Instanzen in der Population berücksichtigen
    - überdurchschnittlische Fitness + kleine Ordnung + kurze Distanz => kommt weiter
      - „Bausteine“
  - Täuschungsproblem
    - Beste Lösung setzt sich nicht aus Bausteinen zusammen
    - Keine Annäherung durch lokale Suche möglich
    - Avg Fitness des Schemas „täuscht“ schlecht Fitness vor
      - obwohl bester Kandidat von Schema abstammt
- Memetische Algorithmen
  - Kulturelle Evolution
    - z.B. Mode
      - Sieht aus wie GA, is aber kulturelle Evolution
    - Erfahrungen und Lernprozesse prägen die Fitness
    - Meist schneller
  - GA + lokale Suche
    - Elemente haben Nachbarschaftsbeziehungen
    - Suche mit gewisser Teife
    - Lokales Maximum
  - Algorithmen
    - Lamarckischer memetischer Algorithmus
      - Gefundene Verbesserungen (durch lernen / lokale Suche) vom Phänotyp in Genotyp schreiben
    - Baldwinischer memetischer Algorithmus
      - Zurückschreiben nicht möglich
      - Verbesserungen erhöhen lediglich die Fitness
- Koevolution
  - Fitness relativ zu anderen Kandidaten
  - Koevolutionsalgorithmen (KEA)
    - Fitness is subjektiv
      - Jeder mit jedem
        - Rechenintensiv
      - Jeder gegen den jeweils besten
    - Mehrere Populationen
    - Konkurrierend (competitive)
      - Wettrüsten
    - Kooperativ (cooperative)
      - Lösungen bestehen aus mehreren Kandidaten
      - Kandidaten als Teillösungen
  - Oszilieren
    - Schwingen der Fitness
    - Vor allem bei intransitiven Problemen
    - Rote Königin Effekt
      - Selektionswahrscheinlichkeit erhöht sich nicht (obwohl verbessert), da auch andere besser werden
      - Wettrüsten ist nicht von Oszilieren zu unterscheiden
- Kodierung
  - Bestimmt Erfolg
  - Suchraum abdecken
    - Möglichst ungültige Lösungen ausschließen
  - Z.B. Bitstring
  - Linkage Problem
    - Hamming-Distanz
      - Anzahl der Bits, an denen sich 2 Zahlen unterscheiden
      - Bsp: 1 und 2 = 01 und 10 => 2 Bits Unterschied
    - Gray-Code
      - Benachbarte Codewörter mit Hamming = 1
    - Mapped multiparameter coding
      - Vector von Elementen, welche einzeln Kodiert sind
      - Kokatinieren der Elemente
    - Adaptive Kodierung
      - Inversionsoperator
        - Funktional zusammenhängende Bits werden getrennt (Crossover)
        - Epistasie ähnliche Effekte
        - Crossover, aber Funktionalität beibehalten
        - Indexe für die Bits
        - {(1,0), (2,0), (3,1)}
        - Bei Phänotyp wieder Rückgängig machen
        - Nach Crossover 1/mehrere Indexe. Was nun?
          - Lösung
            - Crossover nur bei gleicher Permutation
            - Master/Slave: Slave übernimmt Permutation vor Crossover. Danach wieder Rückgängig machen
      - Hot-Spots
        - Ausrufezeichen-Schreibweise
        - 00!110!
        - An ALLEN ! beider Kandidaten auftrennen und Crossover durchführen
          - ! werden wie normale Gene mit getauscht
      - Messy GA
        - Lösungskandidat hat nicht alle/ doppelte Gene
        - Fitness berechnen?
          - Unter-/Überspezifiziert
          - Überspezifiziert
            - First-come-first-serve
          - Unterspezifiziert
            - Mehrmals Zufallswerte
              - Zu hohe Varianz
            - Lokales Maximum berechnen
              - Max als fehlendes Gen einsetzen
    - Epistasie
      - Veränderung überlagert / überdeckt die Wirkung anderer Gene
      - Beeinflusst die Fitness stark
      - Messen:
        - Minimale Edit-Distanz
          - Wie oft muss ich editieren?
          - Distanzmatrix
  - Permutationen (Rundreisen) Kodieren
    - Pfadkodierung
      - Bsp: 1->4->3->2->1
      - (1,4,3,2)
    - Kantenkodierung
      - Basiert auf Index
      - Index -> Zahl
      - (4,1,2,3)
    - Ordinalkodierung
      - Basiert auf Referenz-Rundreise
      - Index der gesuchten Stadt in der Referenz-Rundreise
      - Bsp-Referenz: 1->2->3->4
        - Stadt 1: Ref (1,2,3,4) => (1)
        - Stadt 4: Ref (2,3,4) => (1,3)
        - Stadt 3: Ref (2,3) => (1,3,2)
        - Stadt 2: Ref (2) => (1,3,2,1)
  - Ungültige Lösungen
    - Bessere Kodierung
    - Reparieren beim dekodieren
    - Fitness niedrig => Aussterben
    - Operationen anpassen
      - z.B. gültige Mutationen
- Operatoren
  - Operatoren arbeiten auf Genotyp
  - Genotyp -> Phänotyp -> Fitness
  - Näherungslösung, nicht optimale Lösung
  - Problemspezifische Initialisierung
    - Zusätzlicher Aufwand
    - Wird meist bereits nach wenigen Generationen übertroffen
  - Anfangs: Erkundung (exploration)
    - Ende: Verwertung (exploitation)
  - Selektion
    - Selektionsdruck
      - Wie stark werden bessere bevorzugt
      - Zu stark: Diversität nimmt ab
      - Zu niedrig: Schlechte Kandidaten (aber großer Suchraum)
      - Sinnvoll:
        - Anfang: niedrig
          - Exploration
          - Suchraum verteilen
        - Ende: gering
          - Exploitation
          - Maximum rausholen
    - Varianten
      - Fitnesspropotional
      - Rangbasiert
        - Rangreihenfolge nach Fitness sortiert
        - Verhindert zu viele schlechte Nachkommen
        - 1-n Ränge
      - Lineare rangbasierte Selektion
        - Min und Max Anzahl der Nachkommen
      - Tunierselektion
        - Tunier zwischen k (2 <= k <= n) Kandidaten
        - Je höher k, desto höher is der Selektionsdruck
    - Vorzeitige Konvergenz
      - Varianz am Anfang sehr hoch: Gute Kandidaten werden zu häufig gewählt (Fitnesspropotional)
      - Verdrängt die anderen
      - Suchraum nicht genügend erforscht
      - Dominanzproblem
    - Geschwindigkeit der Evolution hängt von der Varianz ab
    - Skalieren der proportionalen Selektion
      - Lineare Skalierung
        - f'avg = favg
        - f'max = f'avg * Cmult
          - Cmult zwischen 1,2 und 2
        - f' = a * f + b
        - Negativ = 0
        - Beschränkt die Dominanz vom besten Kandidaten
      - Sigma-Skalierung
        - Varianz
        - f' = max(f-(Favg-c * σ), 0)
          - c zwischen 0 und 3
      - Erwartungswert-Modell
        - Beschränkt die Max Nachkommen
        - Immer bei Auswahl einen Wert (0,5 oder 1) abziehen
    - Elitismus
      - Besten/Beste Kandidaten auf jeden Fall mitnehmen
      - Crossover/Mutation können auch ausgeschlossen werden
  - Mutation
    - Bitstrings
    - Flippen 0->1 / 1->0
    - Int/Gleitkomma
      - Random Resetting
      - Creep Mutation
        - um 0 zentrierte Normalverteilung
  - Crossover
    - N-Punkt Crossover
      - Positionsabhängige Verzerrung
        - Vor allem bei Ungeraden N
          - Auftrennen der Gengruppe, wenn Anfang und Ende
        - Nicht bei uniformen Crossover
          - Verteilungsunabhängig Verzerrung
          - Binominalverteilt
    - Uniformes Crossover
      - Wahrscheinlichkeit (pc) von 1 oder 2 abzustammen
    - Diagonal / Multi-Parent Crossover
      - N-Eltern => N-1 Punkt
      - 3 Eltern => 2 Stellen
    - Arithmetic Recombination
      - Allele mit Faktor f gewichtet berechnen
      - Nk1 = f*A1 + (1-f) * A2
      - Nk2 = f*A2 + (1-f) * A1
      - Als Basis für arithmetische Crossover
        - Simple Arithmetic Recombination
          - Index k ermitteln
          - Von 0 bis k einfach tauschen
          - Rest durch arithmetic Recombination
        - Single Arithmetic Recombination
          - 1 Gen (k) durch arithmetic Recombination
        - Whole Arithmetic Recombination
          - Alle durch arithmetic Recombination
  - Permutationen
    - Mutation
      - Swap (Paarweise Platztausch)
        - Gleiche Allelmenge als Vorraussetzung
      - Insert
        - Verschieb die anderen Gene
      - Scramble
        - Teilstück durch Permutation umstellen
      - Inversion
        - Teilstück auswählen
        - Alle Gene des Teilstücks herumdrehen (invertieren des Teilstücks)
    - Crossover
      - Partially Mapped Crossover (PMX)
        - An k1 und k2 auftrennen
        - MITTE tauschen
        - Rest durch Lookup (auf vorheriger Permutation) auffüllen
      - Order Crossover
        - An k1 und k2 auftrennen
        - MITTE beibehalten
        - RECHTS->LINKS befüllen mit jeweils dem anderen RECHTS->LINKS
          - Elemente, die bereits in MITTE sind, überspringen
      - Cycle Crossover
        - Ansatz: Absolute Position der Gene beibehalten
        - In Zyklen auftrennen
        - Zyklen abwechselnd tauschen / behalten