Mehrkriterielle Optimierung 10.4.2012

Carsten Franke

Mehr-kriterielle Evolutionäre Algorihtmen (Beispiele)

- Nicht-elitäre Algorithmen
  - Vector Evaluated Genetic Algorithm (VEGA)
  - Non-dominated Sorting Genetic Algorithm (NSGA)
  - Predator-Prey Evolution Strategy
  - …. sehr viele andere
- Elitäre Algorithmen
  - Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)
  - Strength Pareto Evolutionary Algorithm (SPEA 2)
  - S metric selection Multi-objective Evolutionary Algorithm (SMS-EMOA)
  - Indicator Based Evolutionary Algorithm (IBEA)
  - …. viele andere

Non-Dominated Sorting Genetic Algorithm (NSGA)

- Publiziert 1994 von Srinivas und Deb
- Häufigster nicht-elitärer mehrkriterieller Evolutionärer Algorithmus
- Diversität wird verbessert durch ein „Teilungsschema“ benachbarter Lösungen

NSGA - Algorithmus

Algorithm 3.4 Non-Dominated Sorting Genetic Algorithm (NSGA).

$P_{0,\mu} \leftarrow$ initialization, $P_{0,\mu} \in \mathcal{M}_\mu(\mathbb{I})$;
for $(i = 1$ to $k)$ **do**
  $P_{0,\mu} \leftarrow$ evaluate objective function f_i;
end for
$P_{0,\mu} \leftarrow$ NSGA fitness assignment {See Algorithm 3.5};
$t \leftarrow 0$;
for $(t = 0$ to (Number of Generations - 1)) **do**
  $P'_{t,\lambda} \leftarrow$ fitness proportional selection$(P_{t,\mu}), P'_{t,\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
  $P''_{t,\lambda} \leftarrow$ two point mutation of strategy parameters$(P'_{t,\lambda}), P''_{t,\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
  $P'''_{t,\lambda} \leftarrow$ recombination of object parameters$(P''_{t,\lambda}), P'''_{t,\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
  $P''''_{t,\lambda} \leftarrow$ mutation of object parameters$(P'''_{t,\lambda}), P''''_{t,\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
  for $(i = 1$ to $k)$ **do**
    $P''''_{t,\lambda} \leftarrow$ evaluate objective values f_i;
  end for
  $P''''_{t,\lambda} \leftarrow$ NSGA fitness assignment {See Algorithm 3.5};
  $P_{(t+1),\mu} \leftarrow$ select$(P''''_{t,\lambda}), P_{(t+1),\mu} \in \mathcal{M}_\mu(\mathbb{I})$;
  $t \leftarrow t + 1$;
end for

Anmerkung: oft gilt $\lambda = \mu$

Übung

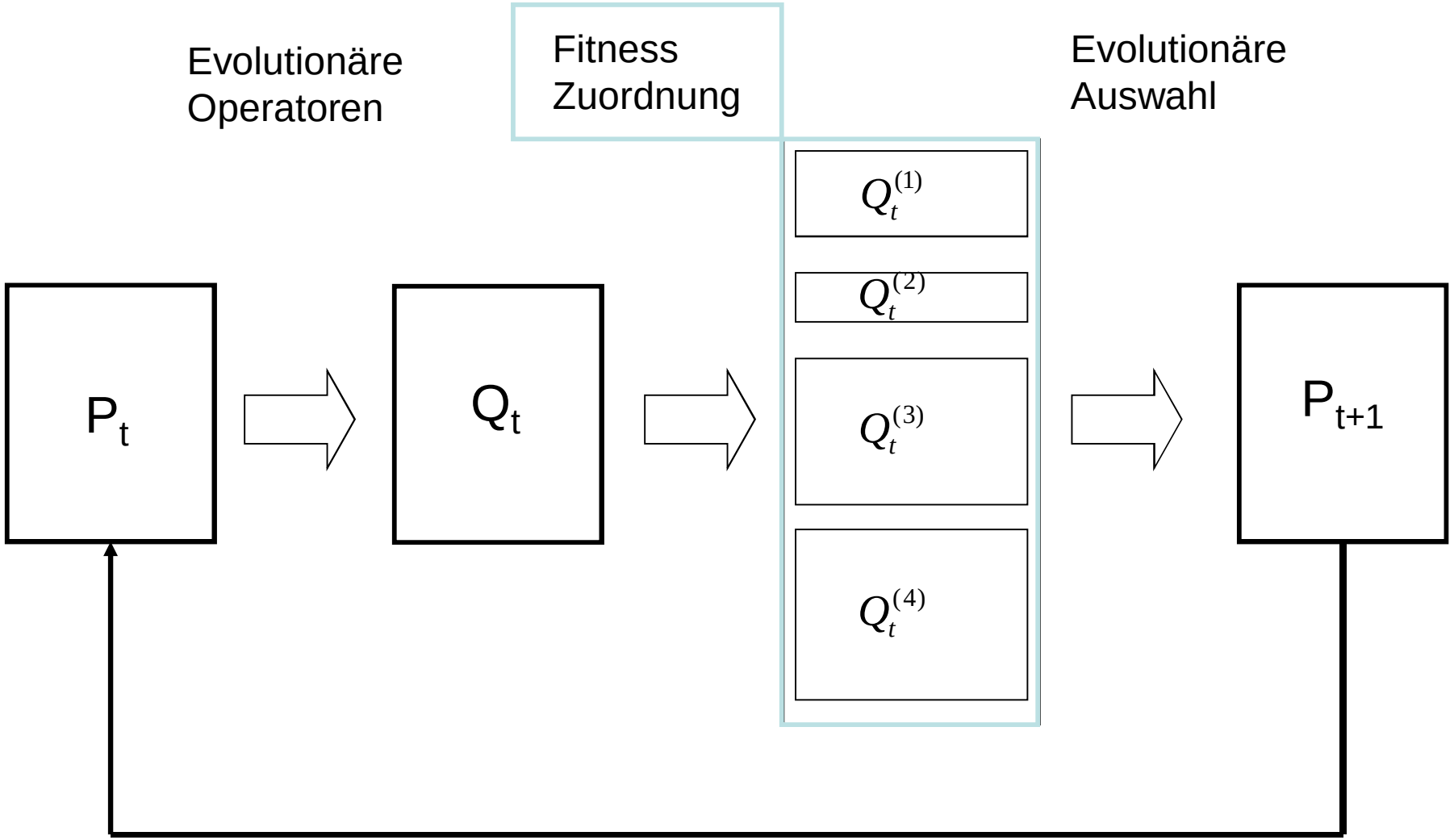
- Welche Gemeinsamkeiten hat eine Evolutionsstrategie und NSGA?
- Ist NSGA ein elitärer Algorithmus? Begründen Sie!

Zwei-Punkte Mutation

- Rechenberg hat diese Mutationsform publiziert
- Funktionsweise ist rein zufällig (r wird zufällig gewählt)

$$\tilde{\sigma} = \begin{cases} \sigma \cdot \alpha, wenn\ r \leq 0{,}5 \\ \sigma / \alpha, wenn\ r > 0{,}5 \end{cases}, \alpha > 1 \text{ und } r \in [0,1[$$

NSGA - Schema


Evolutionäre Operatoren
Fitness Zuordnung
Evolutionäre Auswahl
P_t
Q_t
$Q_t^{(1)}$
$Q_t^{(2)}$
$Q_t^{(3)}$
$Q_t^{(4)}$
P_{t+1}

NSGA Fitness Zuordnung

Algorithm 3.5 NSGA Fitness Assignment.

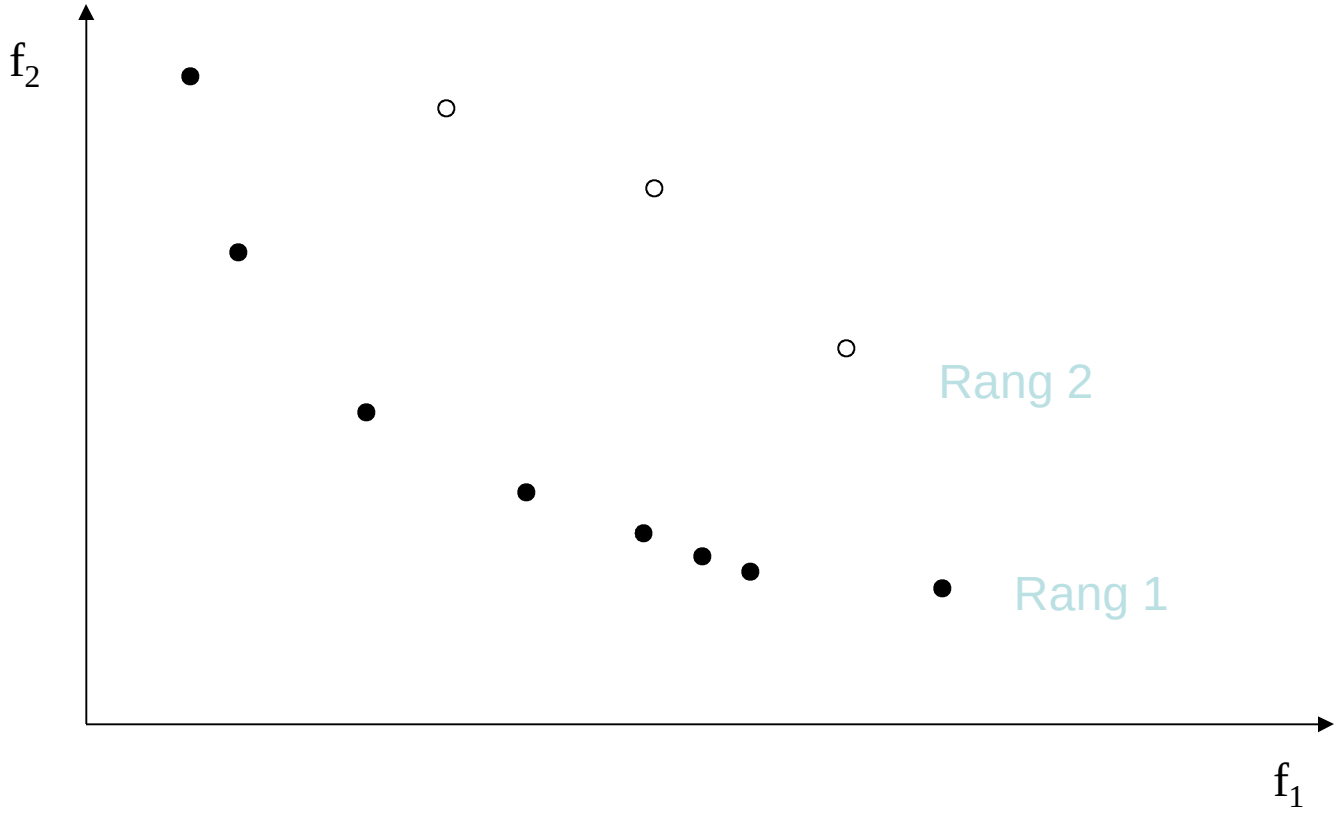
Choose a small positive number ϵ;
$F_{min} = \lambda + \epsilon$;
Classify population P_t according to non-domination: $\left(P_t^{(1)}, P_t^{(2)}, \ldots, P_t^{(\Upsilon)}\right) = Sort\left(P_t, \preceq_p\right)$
{We assume that Υ fronts exist};
for $(j = 1$ to $\Upsilon)$ **do**
  $F_{newMin} = F_{min}$;
  for all $(a \in P_t^{(j)})$ **do**
    Assign rank fitness $F^r(a) = F_{min} - \epsilon$;
    Calculate niche count nc_a among solutions of $P_t^{(j)}$ only; {See Algorithm 3.6.}
    Calculate fitness $F(a) = \frac{F^r(a)}{nc_a}$;
    $F_{newMin} = \min(F(a), F_{newMin})$;
  end for
  $F_{min} = \min(F_{newMin}, F_{min})$;
end for

Übung

- Wozu wird ε genutzt?
- Kann ε > λ sein?

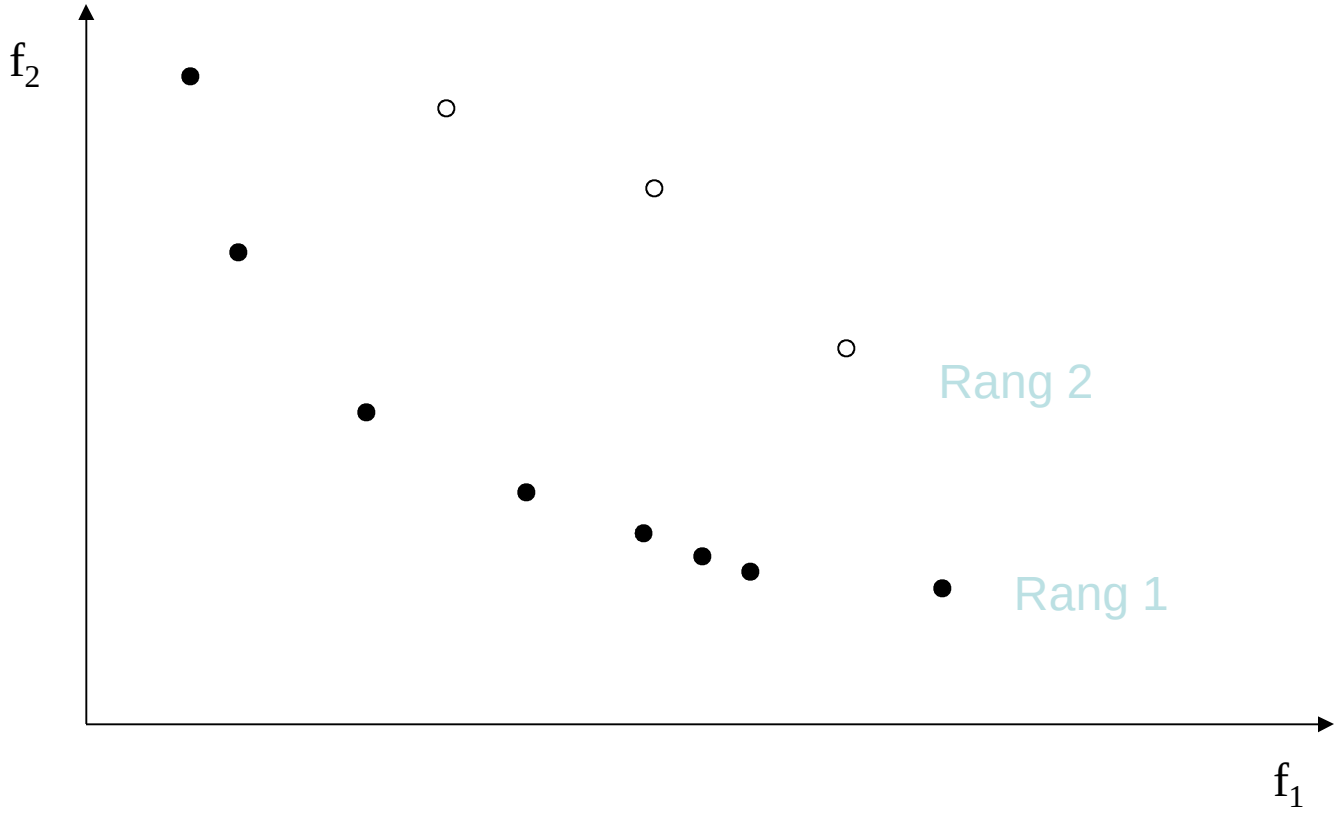
NSGA Fitness-Zuordnung

1. Bestimme den Rang jeder Lösung



NSGA Fitness-Zuordnung

2. Weise F_{min} allen Lösung des Rang 1 zu.

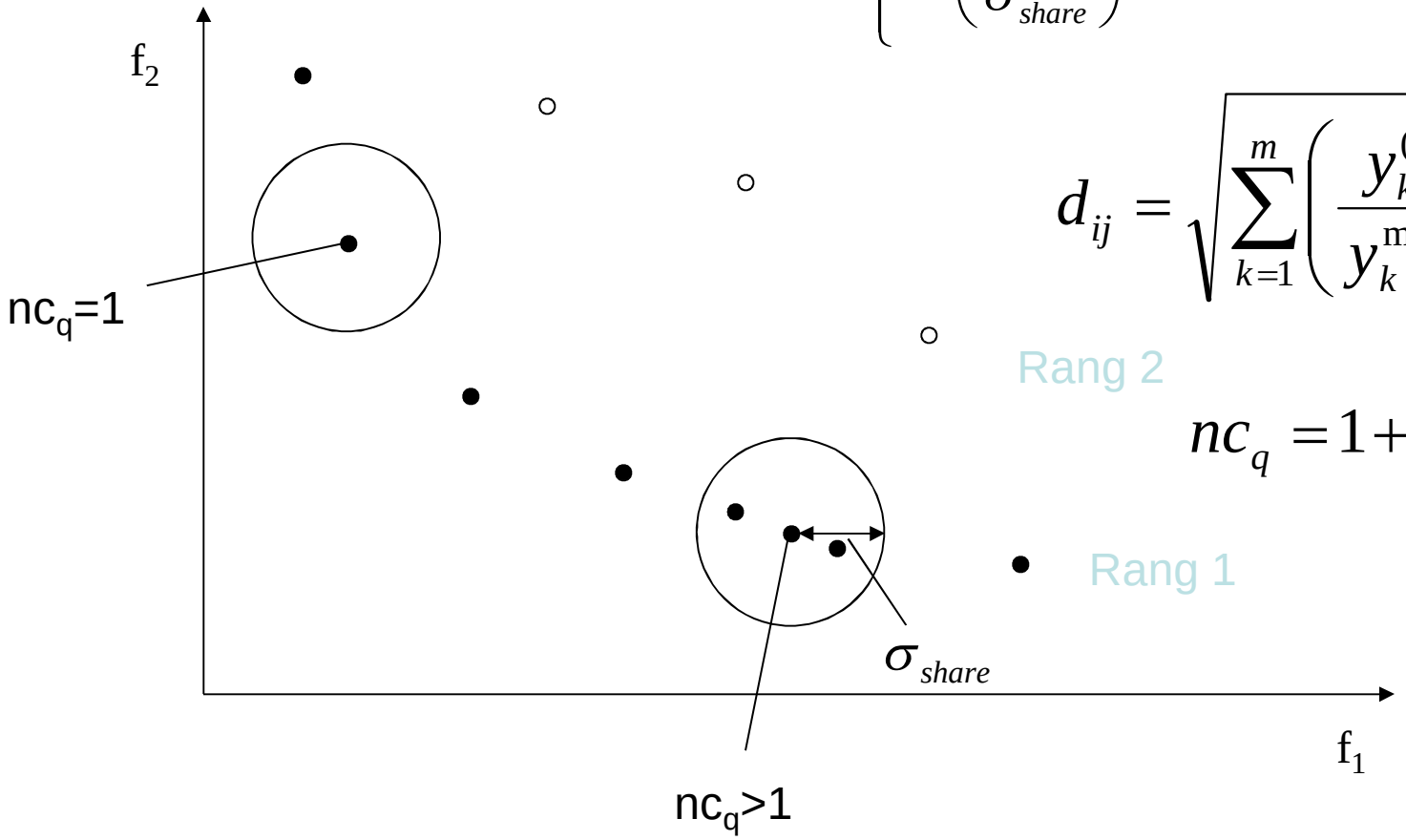


NSGA Fitness-Zuordnung

3. Berechne niche count nc_q für alle Lösungen q mit Rang 1.

$$Sh(d_{ij}) = \left\{ 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^2, if \;\; d_{ij} < \sigma_{share};\; 0 \text{ sonst} \right\}$$

$$d_{ij} = \sqrt{\sum_{k=1}^{m} \left(\frac{y_k^{(i)} - y_k^{(j)}}{y_k^{\max} - y_k^{\min}} \right)^2}$$



$$nc_q = 1 + \sum_{i \in R_t^{(j)}, i \neq q} Sh(d_{iq})$$

Übung

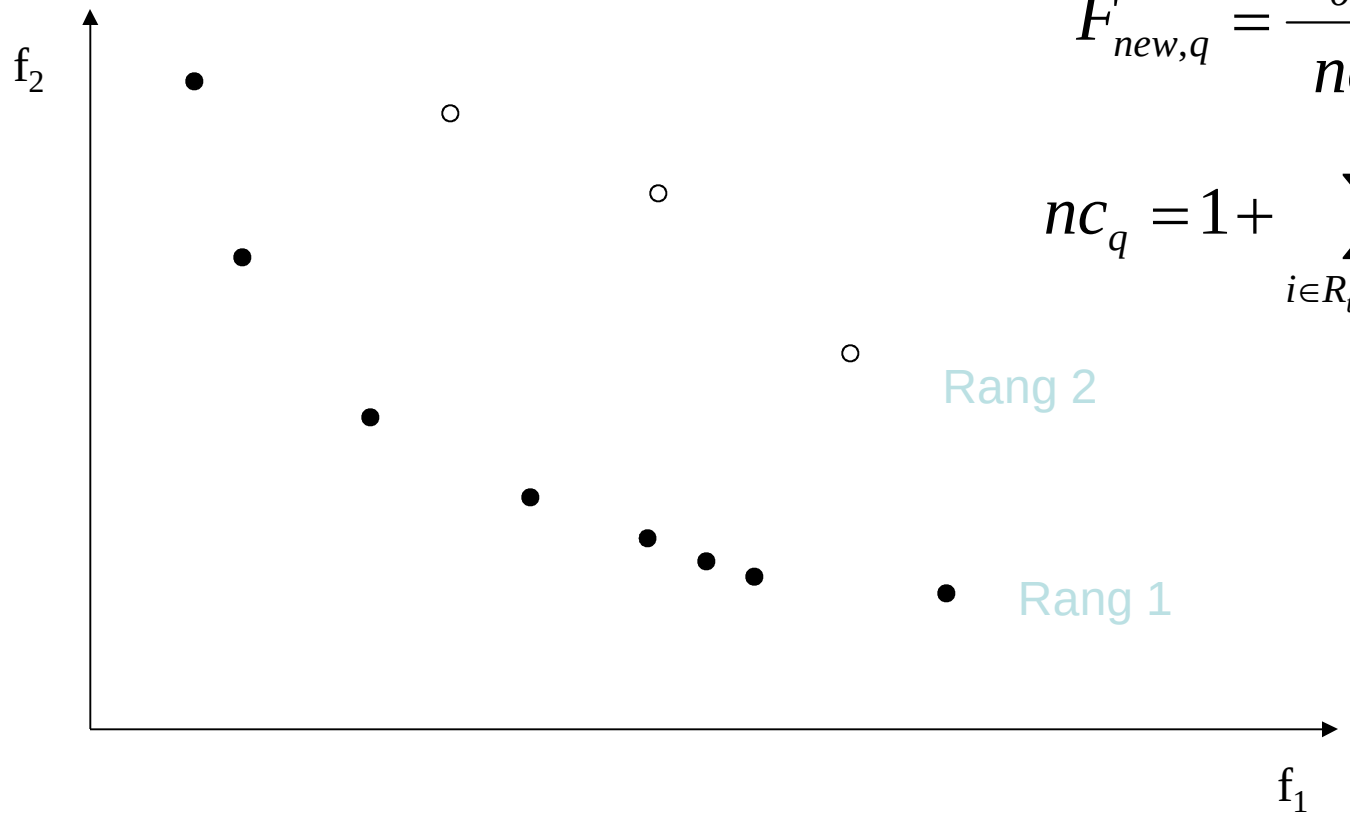
- Geben Sie die kleinste obere Grenze für σ_{share} an!
- Erfolgt die Fitness-Zuordnung auch in der Pareto-Front oder auch in der Pareto-Menge?

NSGA Fitness-Zuordnung

4. Passe alle Fitness-Werte aller Lösungen des gegebenen Rangs an

$$F_{new,q} = \frac{F_{old,q}}{nc_q}$$

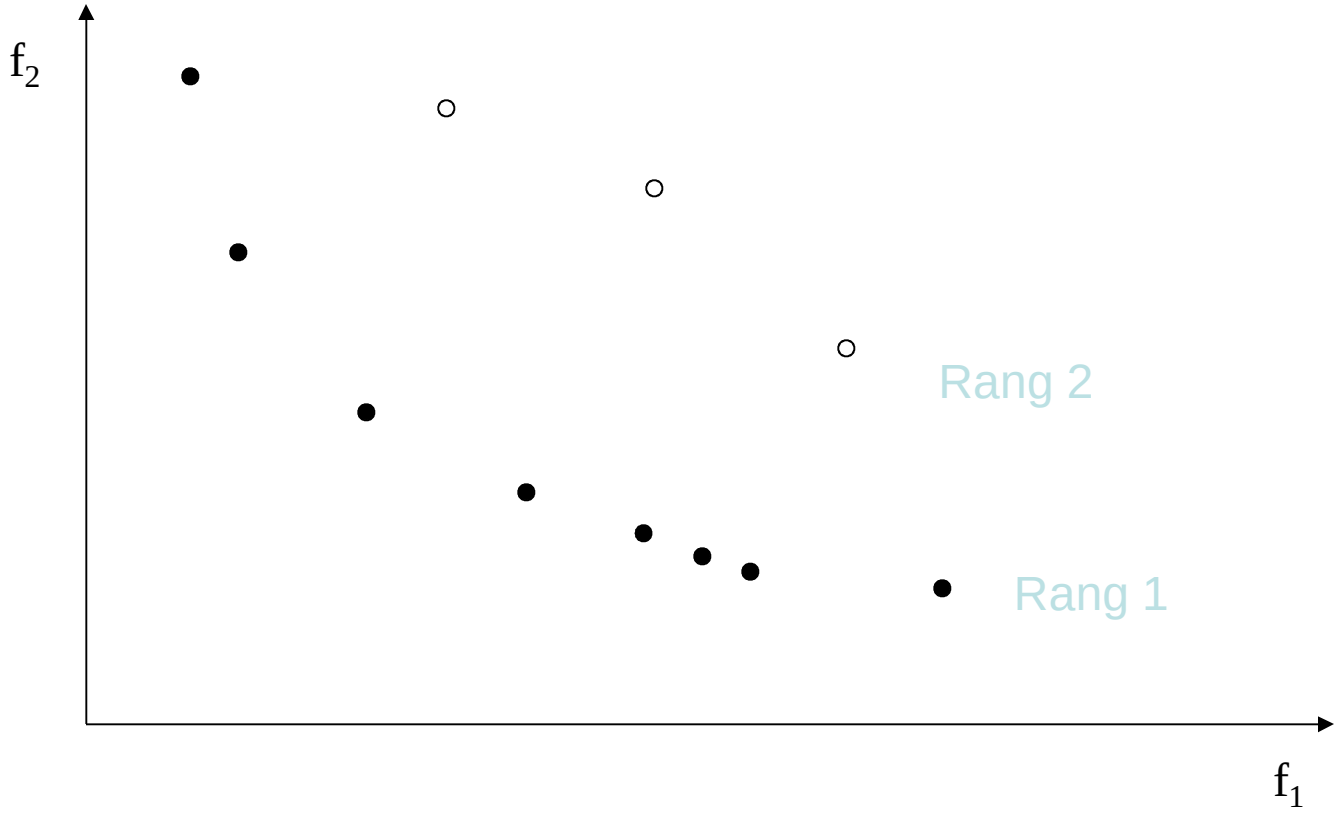
$$nc_q = 1 + \sum_{i \in R_t^{(j)}, i \neq q} Sh(d_{iq})$$



NSGA Zuordnung

5. Weise eine Fitness kleiner als der kleinste Fitnesswerte des gegebenen Rangs den Lösungen des nächsten Rangs zu.

$$F_{nextRank} = \min_{q}(F_{new,q}) - \varepsilon; \quad \varepsilon > 0$$



NSGA Zusammenfassung

- Vorteile
  - Fitness-Zuordnung gemäss des Rangs der nicht-dominierten Front
  - Einführung eines Diversitätserhaltungsoperators (Niche count)
- Nachteile
  - Der Parameter σ_{share} muss vorher definiert sein
  - Nicht-elitärer Algorithmus

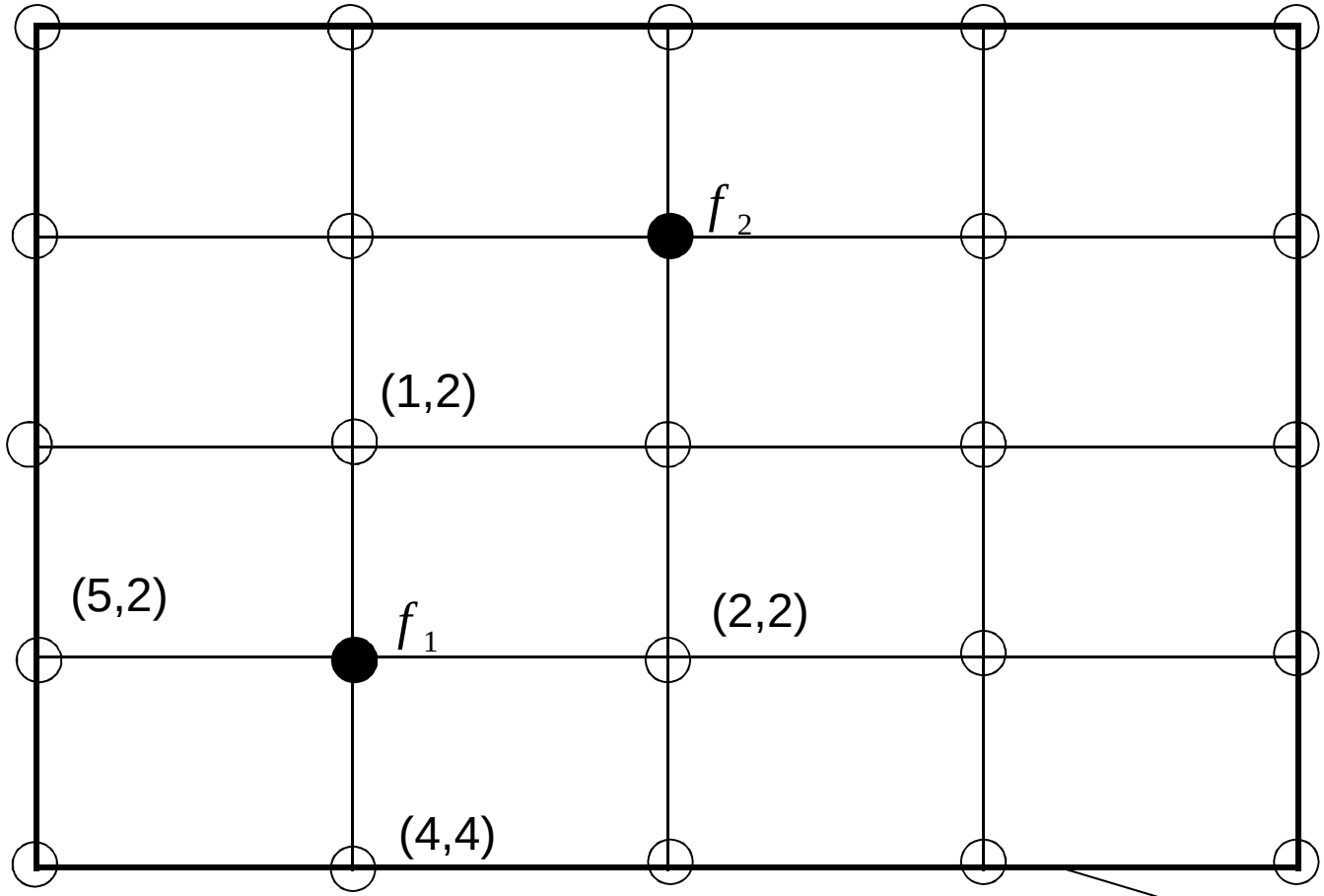
Mehr-kriterielle Evolutionäre Algorihtmen (Beispiele)

- Nicht-elitäre Algorithmen
  - Vector Evaluated Genetic Algorithm (VEGA)
  - Non-dominated Sorting Genetic Algorithm (NSGA)
  - Predator-Prey Evolution Strategy
  - …. sehr viele andere
- Elitäre Algorithmen
  - Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)
  - Strength Pareto Evolutionary Algorithm (SPEA 2)
  - S metric selection Multi-objective Evolutionary Algorithm (SMS-EMOA)
  - Indicator Based Evolutionary Algorithm (IBEA)
  - …. viele andere

Predator-Prey (Räuber-Beute) Evolution Strategy

- Vollkommen anderer Ansatz als herkömmliche Optimierungsmethoden
- Vorgestellt von Laumann 1998
- Verwendet das Konzept der Pareto-Dominanz nicht
- Beute-Objekte repräsentieren die gefundenen Lösungen
- Jäger sind mit einer Optimierungsfunktion assoziiert und versuchen diese zu verbessern

Predator-Prey Initialisierung



ungerichteter Graph

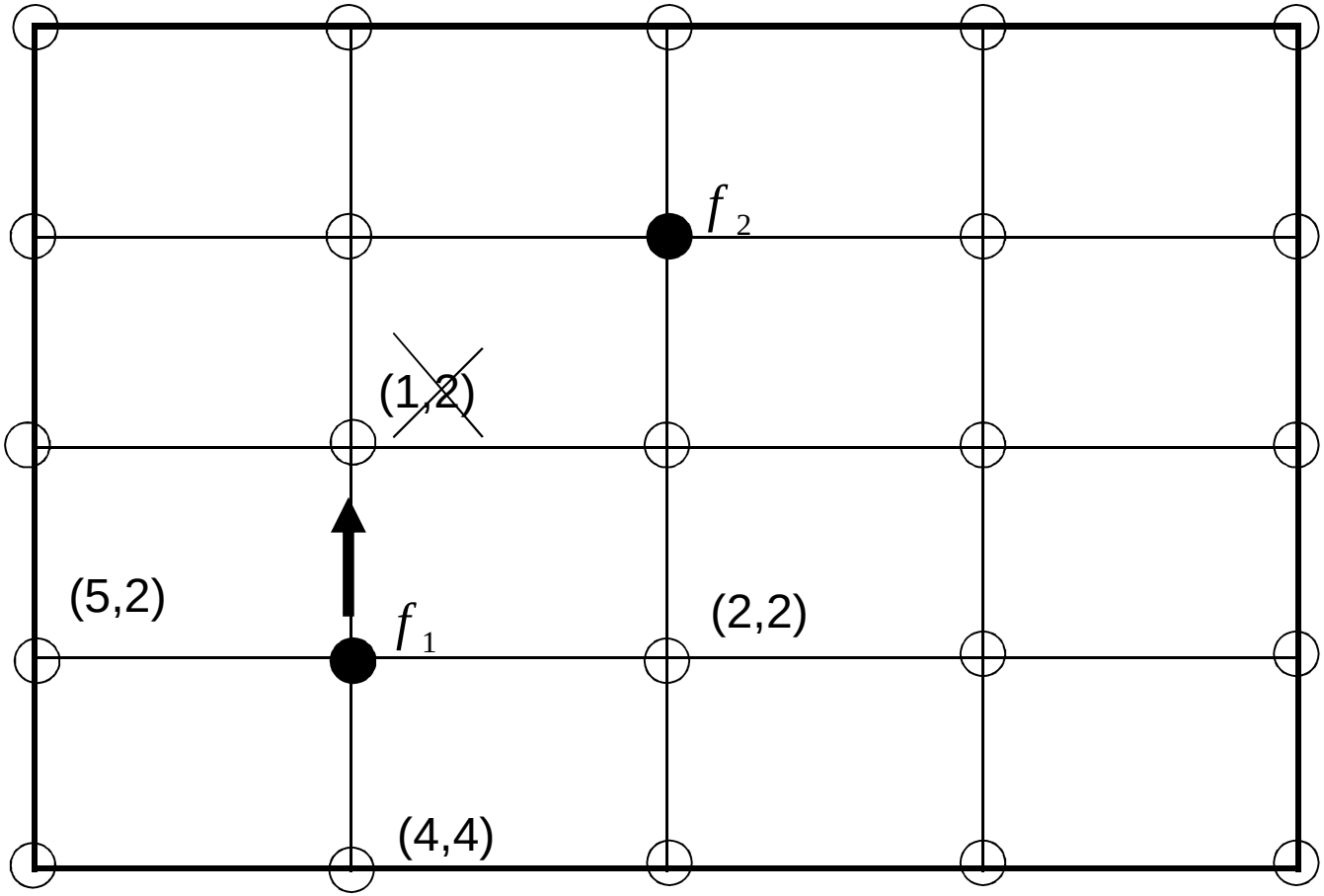
● Jäger (repräsentieren Nutzenfunktionen)

○ Beute (repräsentieren Lösungen)

Übung

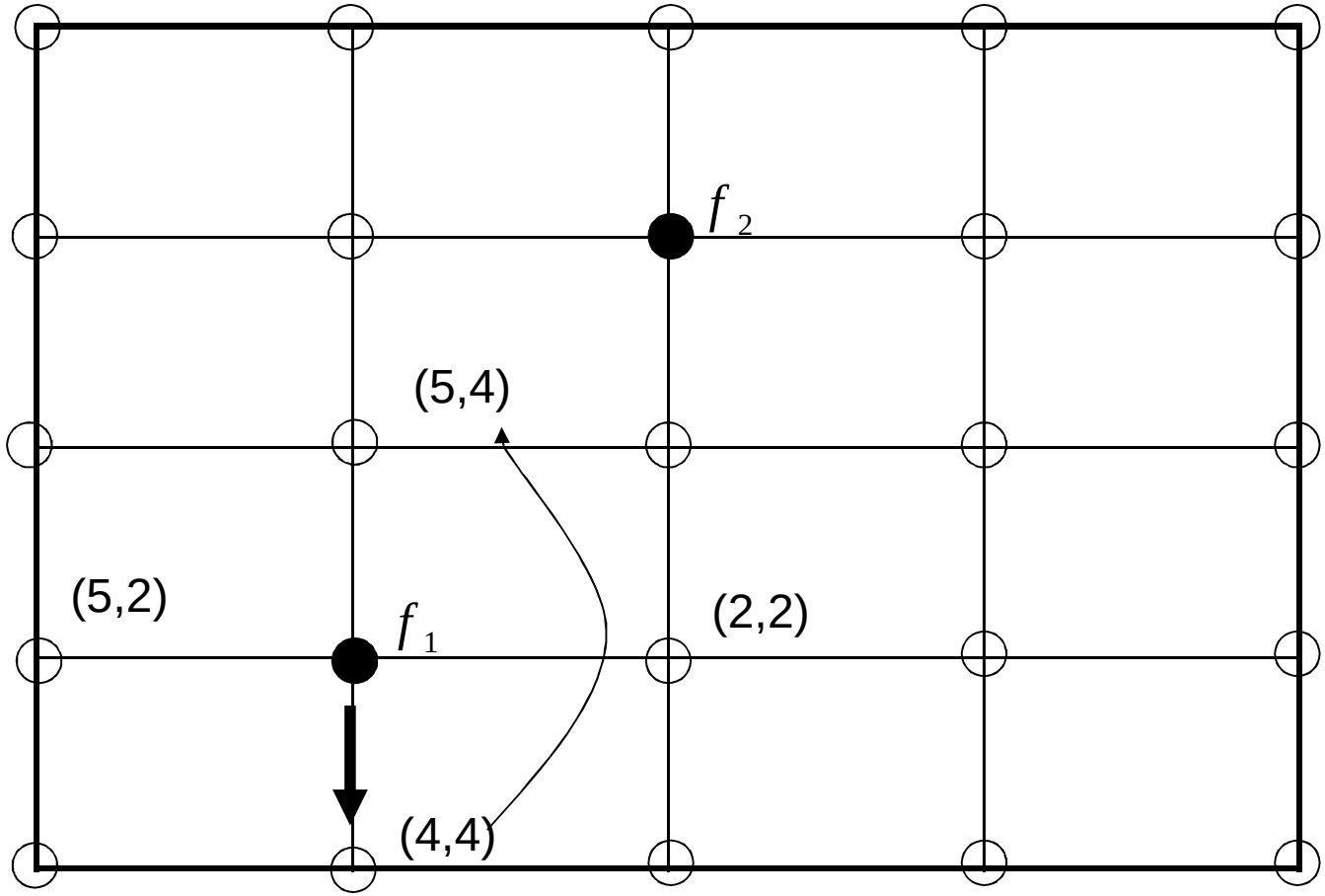
- Definieren Sie “ungerichteter Graph”!
- Worin besteht der Unterschied zu einem Baum?

Predator-Prey Funktionsweise



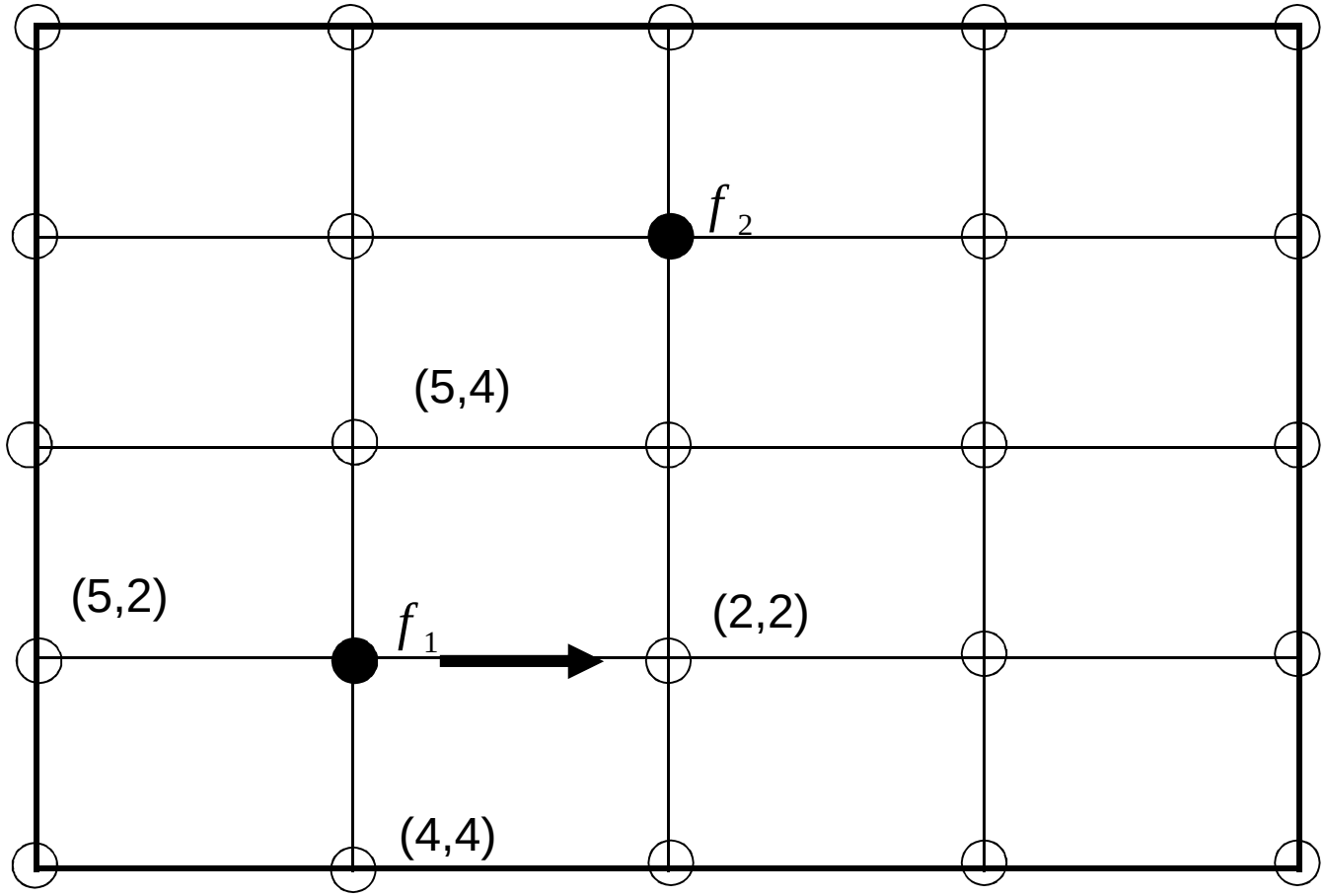
1. Die schlechtestes angrenzende Lösung wird ermittelt (Maximierung) und gelöscht.

Predator-Prey Funktionsweise



2. Eine zufällige andere angrenzende Lösung wird ermittelt und zur Mutation genutzt um die Lösung zu ersetzen.

Predator-Prey Funktionsweise



3. Der Jäger bewegt sich zufällig in eine Richtung.

Predator-Prey Optimierung

- Alle Jäger können sich unabhängig voneinander im Graphen bewegen
- Der Algorithmus erlaubt eine asynchrone Optimierung.
- Der Algorithmus endet in der Regel nach einer vorher festgelegten Anzahl Iterationen.
- Häufig wird die Mutationsstärke am Anfang sehr hoch gewählt und dann pro Iteration um 1% abgesenkt.

Predator-Prey Zusammenfassung

- Vorteile
  - Implementierung ist sehr einfach, da die einzelnen Schritte sehr einfach sind
  - Bevorzugt keine bestimmten Pareto-optimalen Lösungen
  - Kann asynchron ablaufen
- Nachteile
  - Kein direkter Mechanismus der die Pareto-front sicherstellt.
  - Das Verhältnis der Anzahl zwischen Räubern und Beuten ist sehr sensitiv und daher schwer zu wählen

Mehr-kriterielle Evolutionäre Algorihtmen (Beispiele)

- Nicht-elitäre Algorithmen
  - Vector Evaluated Genetic Algorithm (VEGA)
  - Non-dominated Sorting Genetic Algorithm (NSGA)
  - Predator-Prey Evolution Strategy
  - …. sehr viele andere
- Elitäre Algorithmen
  - Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)
  - Strength Pareto Evolutionary Algorithm (SPEA 2)
  - S metric selection Multi-objective Evolutionary Algorithm (SMS-EMOA)
  - Indicator Based Evolutionary Algorithm (IBEA)
  - …. viele andere

Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II)

- Erweiterung von NSGA im Jahre 2000
- Wie bisher werden zunächst Nachkommen durch Reproduktion aus der Elterngeneration erzeugt
- Dann wird aus der Vereinigung beider Mengen entsprechend des Rangs und mittels eines speziellen Operators ausgewählt um die neue Generation zu erstellen

Übung

- Warum war NSGA nicht-elitär?
- Was könnte also zum Beispiel geändert werden um einen elitären Algorithmus zu erzeugen?

NSGA-II Teil 1
Initialisierung

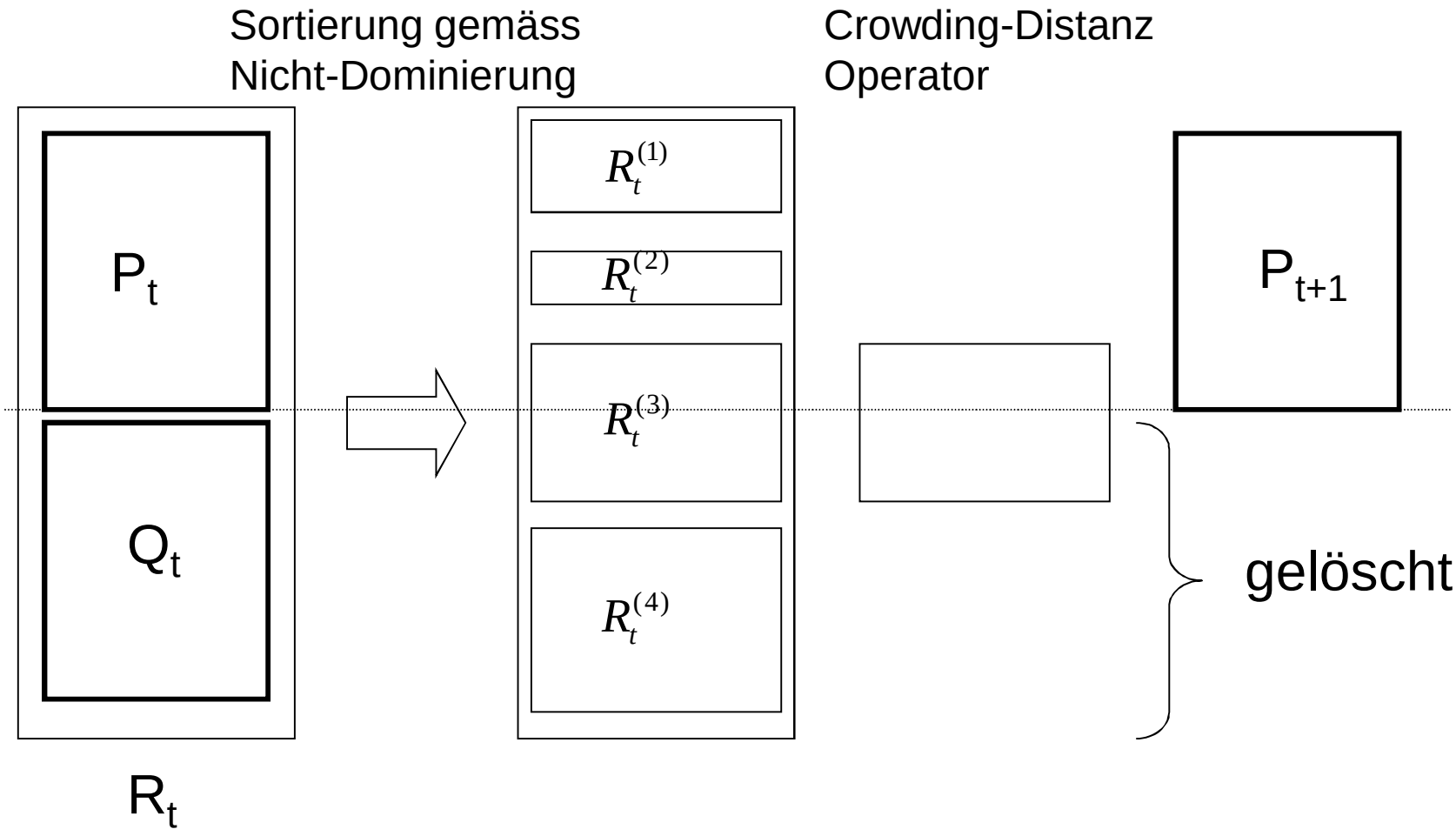
Algorithm 3.7 Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II).

$P_{0,\mu} \leftarrow$ initialization, $P_{0,\mu} \in \mathcal{M}_\mu(\mathbb{I})$;
for $(i = 1$ to $k)$ **do**
  $P_{0,\mu} \leftarrow$ evaluate using objective function f_i;
end for
Assign ranks to all individuals of population $P_{0,\mu}$ according to non-domination;
$P'_{0,\lambda} \leftarrow$ binary tournament selection based on the rank$(P_{0,\mu})$, $P'_{0,\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
$P''_{0,\lambda} \leftarrow$ recombination$(P'_{0,\lambda})$, $P''_{0,\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
$Q_{0,\lambda} \leftarrow$ mutation$(P''_{0,\lambda})$, $Q_{0,\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
for $(i = 1$ to $k)$ **do**
  $Q_{0,\lambda} \leftarrow$ evaluate using objective functionf_i;
end for
$t \leftarrow 0$;

NSGA-II Teil 2

for ($t = 0$ to (Number of Generations-1)) **do**
  $R_{t,\mu+\lambda} = P_{t,\mu} \cup Q_{t,\lambda}$;
  Classify population $R_{t,\mu+\lambda}$ according to non-domination:
  $\left(R_t^{(1)}, R_t^{(2)}, \ldots, R_t^{(\Upsilon)} = Sort(R_{t,\mu+\lambda}, \preceq_p\right)$ {We assume Υ fronts exist};
  $P_{(t+1),\mu} = \emptyset$;
  $c = 1$;
  while $((|P_{(t+1),\mu}| + |R_t^{(c)}| \leq \mu))$ **do**
    crowding distance assignment$(R_t^{(c)})$ {see Algorithm 3.8};
    $P_{(t+1),\mu} = P_{(t+1),\mu} \cup R_t^{(c)}$;
    $c = c + 1$;
  end while
  $Sort(R_t^{(c)}, \preceq_n)$ {sort by crowded comparison operator, see Algorithm 3.9};
  $P_{(t+1),\mu} = P_{(t+1),\mu} \cup R_t^{(c)}[1 : (\mu - |P_{(t+1),\mu}|)]$ {the best $(\mu - |P_{(t+1),\mu}|)$ elements of $R_t^{(c)}$};
  $P'_{(t+1),\lambda} \leftarrow$ binary tournament selection$(P_{(t+1),\mu}), P'_{(t+1),\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
  $P''_{(t+1),\lambda} \leftarrow$ mutation of strategy parameters$(P'_{(t+1),\lambda}), P''_{(t+1),\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
  $P'''_{(t+1),\lambda} \leftarrow$ recombination$(P''_{(t+1),\lambda}), P'''_{(t+1),\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
  $Q_{(t+1),\lambda} \leftarrow$ mutation$(P'''_{(t+1),\lambda}), Q_{(t+1),\lambda} \in \mathcal{M}_\lambda(\mathbb{I})$;
  for ($i = 1$ to k) **do**
    $Q_{(t+1),\lambda} \leftarrow$ evaluate using objective function f_i;
  end for
  $t \leftarrow t + 1$;
end for

Schema für NSGA-II



Crowding-Distanz

Algorithm 3.8 NSGA-II Crowding distance assignment procedure for set R.

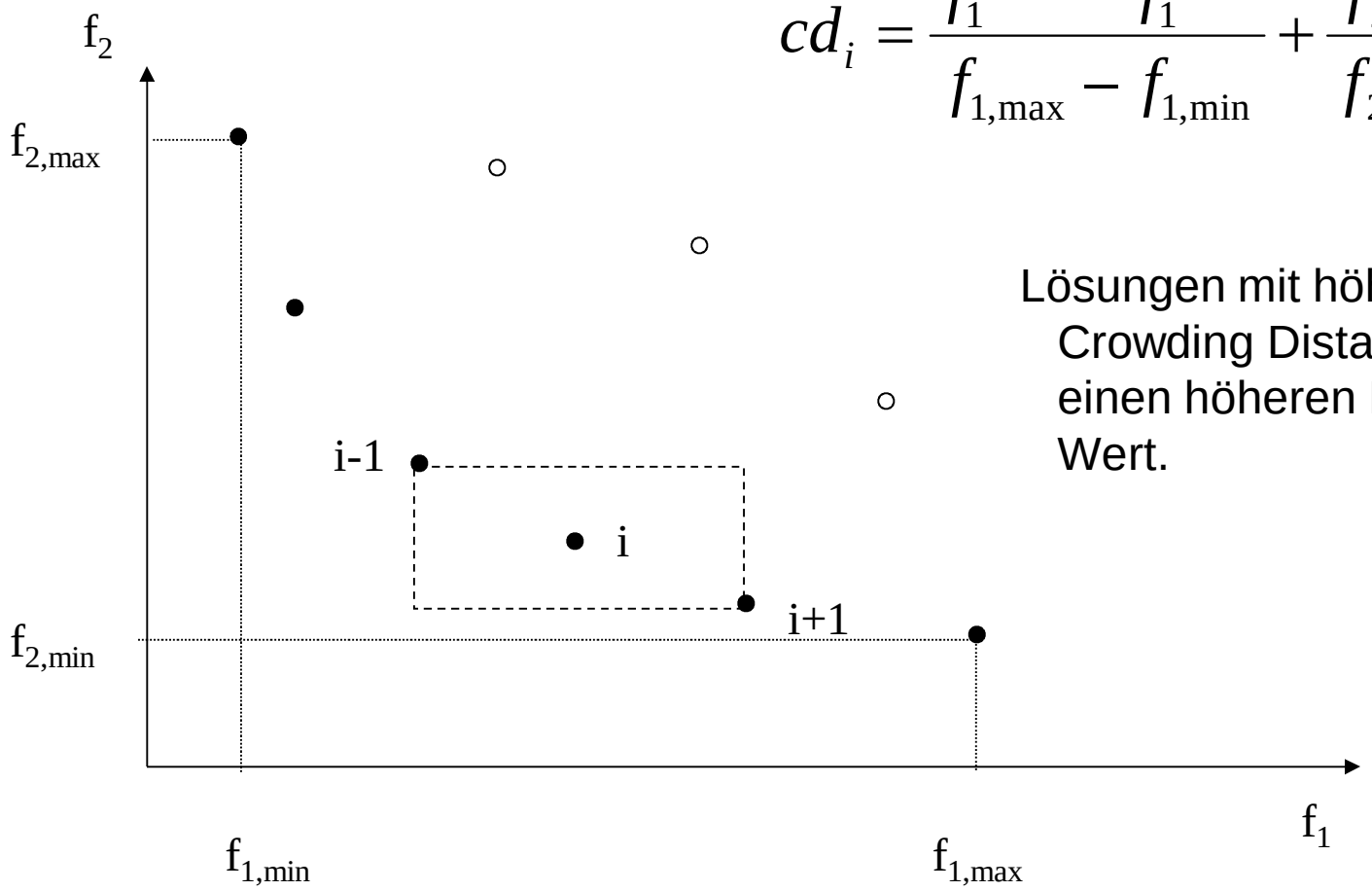
$\Gamma = |R|$ {We assume Γ individuals in set R};
for all $(i \in R)$ **do**
  $R[i]_{distance} = 0$ {for each individual i set the crowding distance to 0};
end for
for $(o = 1$ to $k)$ **do**
  Sort R according to increasing objective function f_o: $(R[1_{f_o}], R[2_{f_o}], \ldots, R[\Gamma_{f_o}]) = Sort(R, o)$;
  $V_{f_o}^{max} = R[\Gamma_{f_o}]_{f_o}$;
  $V_{f_o}^{min} = R[1_{f_o}]_{f_o}$;
  $R[1_{f_o}]_{distance} = R[\Gamma_{f_o}]_{distance} = \infty$ {first and last element have distance ∞};
  for $(j = 2$ to $(\Gamma - 1))$ **do**
    $R[j_{f_o}]_{distance} = R[j_{f_o}]_{distance} + \frac{R[(j-1)_{f_o}]_{f_o} - R[(j+1)_{f_o}]_{f_o}}{V_{f_o}^{max} - V_{f_o}^{min}}$;
  end for
end for

Übung

- Warum haben die Lösungen mit den minimalen und maximalen Werten eine unendliche Crowding-Distanz?

Crowding Distanze Berechnung

$$cd_i = \frac{f_1^{(i+1)} - f_1^{(i-1)}}{f_{1,\max} - f_{1,\min}} + \frac{f_2^{(i-1)} - f_2^{(i+1)}}{f_{2,\max} - f_{2,\min}}$$



Lösungen mit höherer Crowding Distanz haben einen höheren Fitness-Wert.

Vergleichsoperator

Algorithm 3.9 NSGA-II Crowded Comparison Operator $\preceq_n$ for two individuals a_1 and a_2.

$a_{1,rank}$ denotes the rank of i;
$a_{1,distance}$ denotes the crowding distance, see Algorithm 3.8;
if $(a_{1,rank} < a_{2,rank})$ **then**
  $a_1 \preceq_n a_2$
else if $(a_{2,rank} < a_{1,rank})$ **then**
  $a_2 \preceq_n a_1$
else if $(a_{1,distance} > a_{2,distance})$ **then**
  $a_1 \preceq_n a_2$
else
  $a_2 \preceq_n a_1$
end if

NSGA-II Zusammenfassung

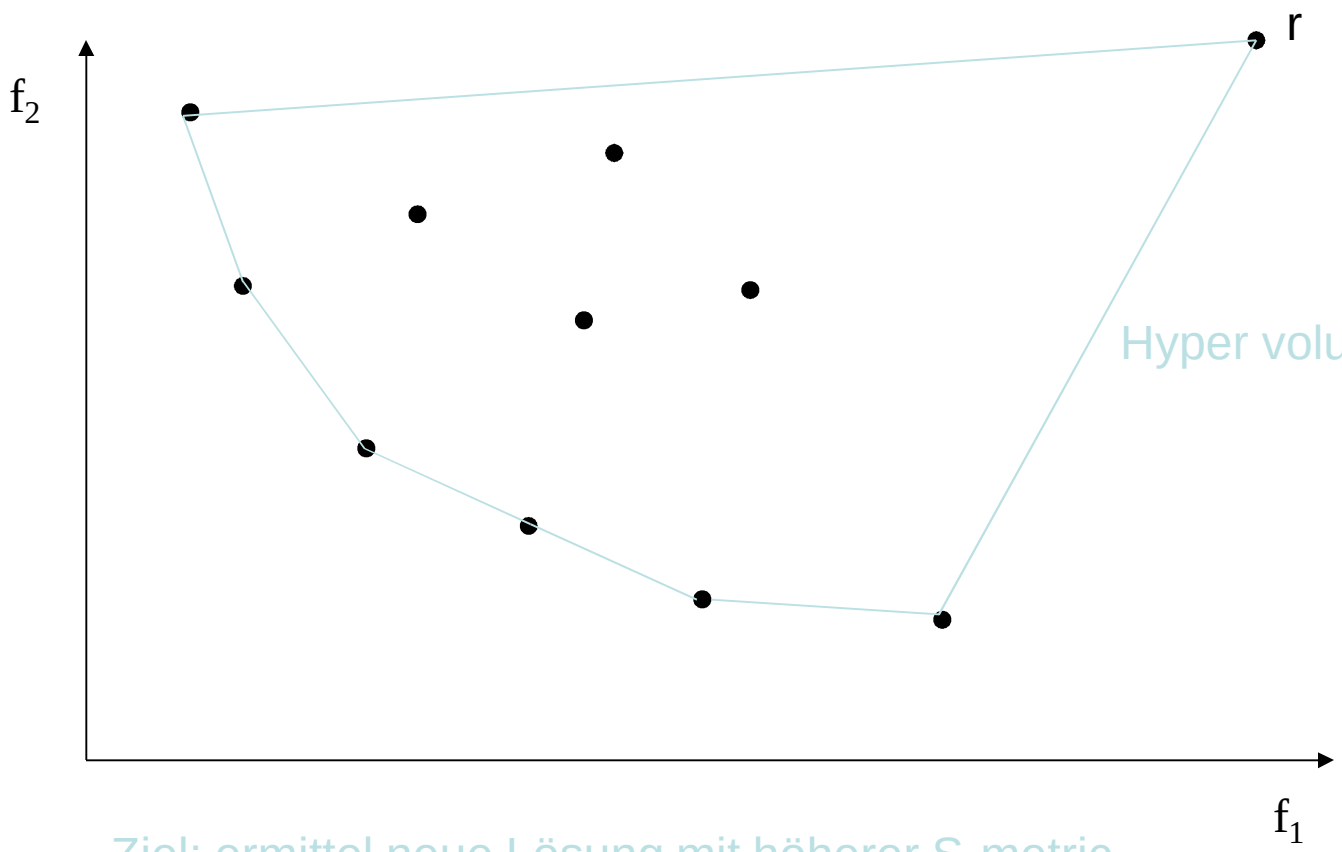
- Vorteile
  - Der Niche-Parameter von NSGA wurde vermieden
  - Turnierselektion sorgt für elitäre Lösungen
- Nachteile
  - Funktioniert nicht sonderlich gut in praktischen Problemfällen mit höheren Dimensionen

Mehr-kriterielle Evolutionäre Algorihtmen (Beispiele)

- Nicht-elitäre Algorithmen
  - Vector Evaluated Genetic Algorithm (VEGA)
  - Non-dominated Sorting Genetic Algorithm (NSGA)
  - Predator-Prey Evolution Strategy
  - …. sehr viele andere
- Elitäre Algorithmen
  - Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)
  - Strength Pareto Evolutionary Algorithm (SPEA 2)
  - S metric selection Multi-objective Evolutionary Algorithm (SMS-EMOA)
  - Indicator Based Evolutionary Algorithm (IBEA)
  - …. viele andere

Mehrkriterielle Optimierung mit Evolutionären Algorithmen mittels S-metric

r – wird automatisch bestimmt



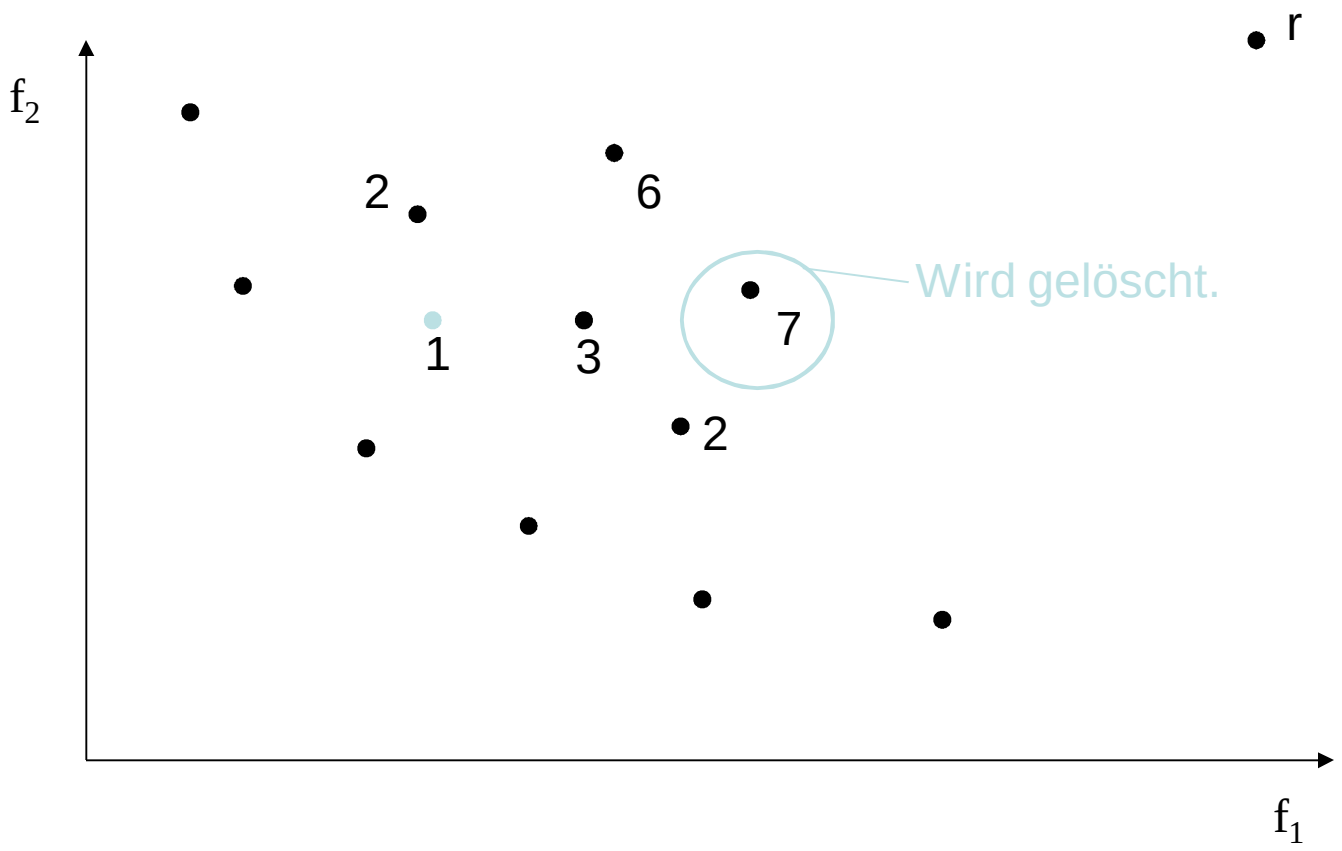
Hyper volume = S-metric

Ziel: ermittel neue Lösung mit höherer S-metric.

Berechnung der S-metric: $O(n \log n + n^{d/2})$

(μ+1) Algorithmus

Fall 1: Eliminiere das Individuum mit höchstem Dominanzwert.



(μ+1) algorithm

Fall 2: Eliminiere das Individuum mit kleinstem S-metric Beitrag.

Algorithmus wird nach einer fest deminierten Generationszahl beendet.

Vorlesungsplanung

- 21.02.2012: Einkriterielle Evolutionäre Optimierung I (CF)
- 28.02.2012: Einkriterielle Evolutionäre Optimierung II (CF)
- 06.03.2012: Test (1+2), Mehrkriterielle Evolutionäre Optimierung I (CF)
- 13.03.2012: Statistische Lerntheorie I (JP)
- 20.03.2012: Statistische Lerntheorie II (JP)
- 27.03.2012: Test (4+5), Neuronale Netze (JP)
- 10.04.2012: Mehrkriterielle Evolutionäre Optimierung II (CF)
- 08.05.2012: Genetische Fuzzy Systeme (CF)
- 09.05.2012: Test (3+7+8), Simulated Annealing und andere Suchmethoden (CF)
- 15.05.2012: Meta-Heuristiken (ACO, PSO) (CF)
- 22.05.2012: Support Vector Maschinen I (JP)
- 29.05.2012: Support Vector Maschinen II (JP)
- 05.06.2012: Test (6+7+12), Clustering (JP)
- 12.06.2012: Lernen und Spieltheorie (JP)
- 26.06.2012: 1. Termin mündliche Prüfungen
- 03.07.2012: 2. Termin mündliche Prüfungen