UNIKASSEL
VERSITÄT

SOFTWARE ENGINEERING
UNIVERSITÄT KASSEL

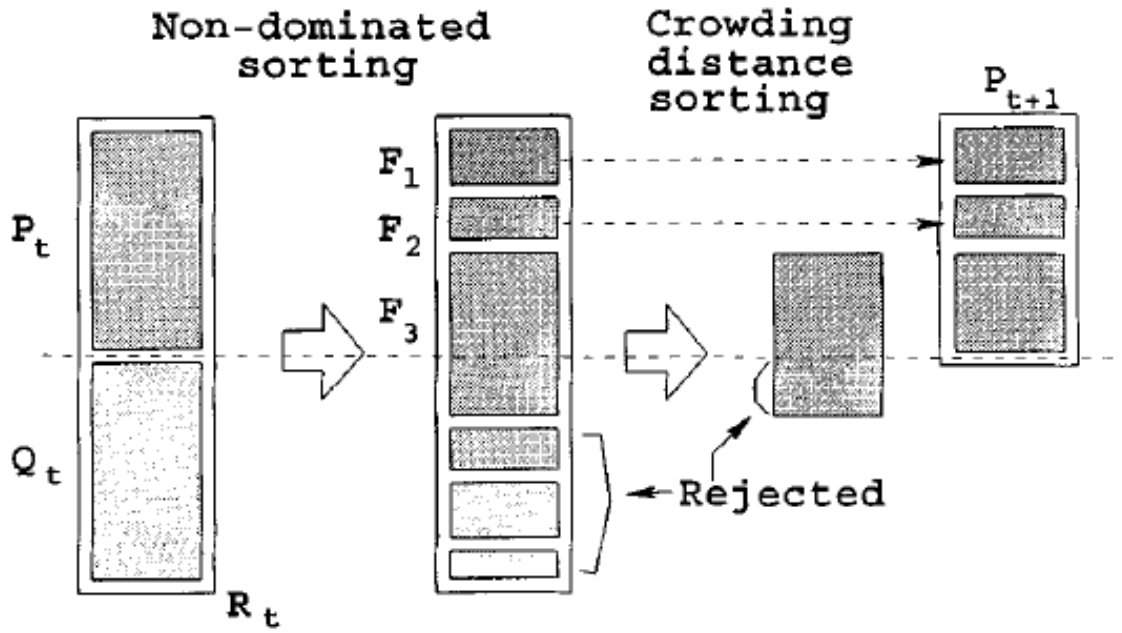
Multikriterielle evolutionäre Algorithmen: NSGA-II und SPEA2

Jens Kosiol

Ein evolutionärer Algorithmus für die multi-kriterielle Optimierung: NSGA-II

- NSGA-II steht für nondominated sorting genetic algorithm II
- Veröffentlicht 2000 von Deb et al. (Zeitschriftenartikel: [DPAM02]); das Paper wurde 53751 mal zitiert (laut GoogleScholar; Stand: 27.01.2024)
- Eigenschaften:
  - Elitär
  - Schneller Sortieralgorithmus in Bezug auf den Rang in der Pareto-Ordnung
  - Parameterfreier Mechanismus zur Sicherstellung von Diversität
  - Unterstützt Nebenbedingungen

Grunddesign NSGA-II



Quelle: [DPAM02]

Grundlegendes Design:

- NSGA-II ist elitär: $(\mu + \mu)$-Selektion wird verwendet (P_t ist die Population der t-ten Iteration, Q_t die in der t-ten Iteration berechneten Nachkommen)
- Lösungen werden gemäß ihres Rangs in der Pareto-Ordnung sortiert und nicht-dominierte bzw. selten dominierte Lösungen setzen sich durch. Hierfür wird ein schnelles Sortierverfahren eingesetzt.
- Bei gleichem Rang werden bevorzugt Lösungen in die nächste Population aufgenommen, die in dünn besetzten Teilen des Suchraums liegen. Hierfür wird die crowding distance als Maß verwendet.

Non-domination Rang

Gegeben eine Menge von Individuen M und Fitnessfunktionen $f = (f_1, \dots, f_n)$, dann hat ein Individuum $x \in M$ non-domination Rang 1 (in M), falls es von keinem $y \in M$ Pareto-dominiert wird. Wir notieren die Teilmenge aller Elemente von Rang 1 mit F_1.

Ein Individuum $x \in M$ hat non-domination Rang k (in M), falls es nicht selbst von Rang $1, \dots, k-1$ ist und nur von Individuen der Ränge $1, \dots, k-1$ dominiert wird, also falls x den non-domination Rang 1 in der Menge $M \setminus (F_1 \cup \dots \cup F_{k-1})$ hat.

Die Menge F_i heißt die i-te non-domination Front.

Schnelle Sortierung bzgl. non-domination Rang


```
fast-non-dominated-sort(P)
for each p ∈ P
   S_p = ∅
   n_p = 0
   for each q ∈ P
      if (p ≺ q) then                    If p dominates q
         S_p = S_p ∪ {q}                 Add q to the set of solutions dominated by p
      else if (q ≺ p) then
         n_p = n_p + 1                   Increment the domination counter of p
   if n_p = 0 then                       p belongs to the first front
      p_rank = 1
      F_1 = F_1 ∪ {p}
i = 1                                    Initialize the front counter
while F_i ≠ ∅
   Q = ∅                                 Used to store the members of the next front
   for each p ∈ F_i
      for each q ∈ S_p
         n_q = n_q − 1
         if n_q = 0 then                 q belongs to the next front
            q_rank = i + 1
            Q = Q ∪ {q}
   i = i + 1
   F_i = Q
```

Quelle: [DPAM02]

Eigenschaften:

- Laufzeit in $O(MN^2)$, wobei M die Anzahl der Optimierungsziele ist und N die Größe der zu sortierenden Menge
  - In der For-Schleife finden N^2 Vergleiche jeweils bezüglich der M Optimierungsziele statt
  - In der While-Schleife werden höchstens $N-1$ Elemente jeweils höchstens $(N-1)$-mal besucht, um ihren domination counter zu dekrementieren; also Größenordnung $O(N^2)$
- Speicherplatzbedarf in $O(N^2)$

Diversität per Crowded-Comparison

- Vor NSGA-II haben viele multikriterielle evolutionäre Algorithmen fitness sharing eingesetzt (Gewichtung der Fitness einer Lösung mit der Anzahl der Lösungen in der Umgebung), um Diversität sicherzustellen.
  - Erfolg ist abhängig von passender Wahl des Sharing-Parameters (Radius der zu betrachtenden Lösungen).
  - Zwischen jedem Paar von Lösungen muss die Distanz gemessen werden ($O(N^2)$ Vergleiche für Menge der Größe N).
- Ansatz NSGA-II: Bevorzuge Lösungen, die isoliert stehen.
  - Die crowding distance dient als Tiebreaker zwischen Lösungen gleichen Rangs.
  - Die crowding distance ist Maß der Isolation einer Lösung; der Umfang des Quaders, den die im objective space benachbarten Lösungen gleichen Rangs aufspannen, wird abgeschätzt.

Berechnung Crowding distance

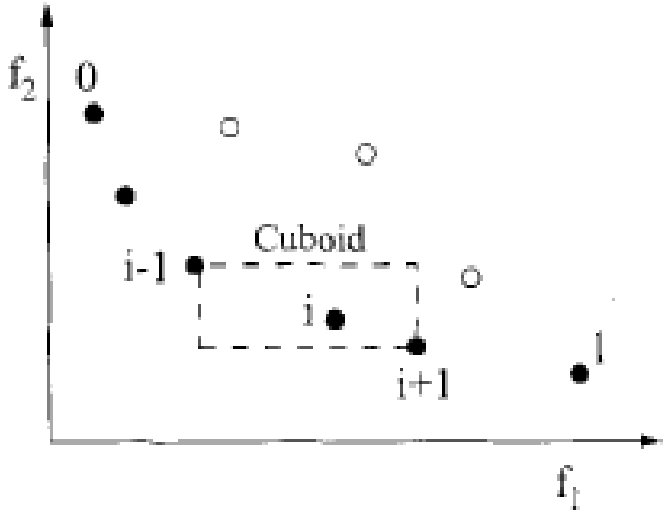
crowding-distance-assignment(I)
k = |I|
for each $1 \leq i \leq k$, set $I[i]_{dist}=0$
for each objective m
  I = sort(I,m)
  $I[1]_{dist}=I[k]_{dist}=\infty$
  for i=2 to (k-1)
    $I[i]_{dist}+=\frac{f_m(I[i+1]-f_m(I[i-1])}{f_m(I[k])-f_m(I[1])}$

Größe der Menge I
Initialisierung der Distanzen

Aufsteigend nach Fitness sortieren
Randfälle erhalten den höchsten Wert

Normalisierte Distanz der Nachbarn

(bezüglich des gerade betrachteten Optimierungsziels) wird addiert



Quellen: [DPAM02]

Wichtige Eigenschaften:

- Laufzeit: $O(mk \log k)$
- parameterfrei

Gesamtaufbau NSGA-II

$R_t = P_t \cup Q_t$
$\mathcal{F} = \text{fast-non-dominated-sort}(R_t)$
$P_{t+1} = \emptyset$ and $i = 1$
until $P_{t+1} + \mathcal{F}_i \leq N$
  crowding-distance-assignment($\mathcal{F}_i$)
  $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
  $i = i + 1$
Sort($\mathcal{F}_i, \prec_n$)
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$
$Q_{t+1} = \text{make-new-pop}(P_{t+1})$

$t = t + 1$

Quelle: [DPAM02]

- Fitness wird über den Crowded-Comparison Operator gemessen: Eine Lösung x ist besser als die Lösung y, falls sie kleineren non-domination Rang hat oder gleichen Rang aber eine höhere Crowding Distance innerhalb der gemeinsamen non-domination Front.
- Parent Selection geschieht über Turniere der Größe 2; Fitnessvergleich auch hier über den Crowded Comparison Operator.
- Survivor Selection geschieht per $(\mu + \mu)$ wie im Algorithmus links illustriert.
- Kann mit beliebigen Variationsoperatoren verwendet werden.

Integration von Nebenbedingungen

Hat das gegebene Optimierungsproblem Nebenbedingungen, kann der Vergleich der Fitness zweier Lösungen auf die folgende Art und Weise erweitert werden: Lösung x dominiert die Lösung y, falls:

1. x ist valide und y nicht;
2. beide Lösungen sind ungültig, aber x verletzt die Nebenbedingungen in einem geringeren Ausmaß;
3. beide Lösungen sind valide und x dominiert y bezüglich des Crowded Comparison Operators.

SPEA2

- SPEA 2 steht für Strength Pareto Evolutionary Algorithm 2
- Veröffentlicht 2001 von Zitzler, Laumanns und Thiele in [ZLT01]; 9999 Zitate laut Google Scholar (Stand: 27.01.2024)
- Grundsätzlich ähnlich wie der NSGA-II aufgebaut, aber Unterschiede in Berechnung der Fitness und Selektion

Grundsätzlicher Aufbau SPEA2

SPEA2 ähnelt in seinen Grundzügen dem NSGA-II stark:

- Parent Selection wird durch Turnier (mit Zurücklegen) mit Turniergröße 2 durchgeführt.
- Survivor Selection wird auch per Plus-Selektion durchgeführt, aber allgemein $(\mu + \lambda)$ erlaubt.
- Variationsoperatoren, Terminationsbedingung und Initialisierung der ersten Population können dem Problem angepasst werden.

Fitnessberechnung im SPEA2

Die Fitness $F(i)$ einer Lösung i besteht aus zwei Komponenten: einer Komponente $R(i)$, die die Pareto-Dominanz berücksichtigt (raw fitness), und einer Komponente $D(i)$, die den Abstand zu weiteren Lösungen berücksichtigt (density).

Gegeben Population P_t und Nachkommen Q_t (in Iteration t) berechnet sich die Fitness als $F(i) = R(i) + D(i)$, wobei:

- $R(i) = \sum_{j \in P_t \cup Q_t, j > i} S(j)$, wobei $S(i) = |\{j \in P_t \cup Q_t \mid i > j\}|$ und $>$ die Pareto-Dominanz bezeichnet;
- $D(i) = \frac{1}{\sigma_i^k + 2}$, wobei σ_i^k den Abstand von i zum k-ten Nachbarn (im objective space) bezeichnet (typischer Wert für k: $\sqrt{\mu + \lambda}$)

Illustration: Berechnung der Fitness

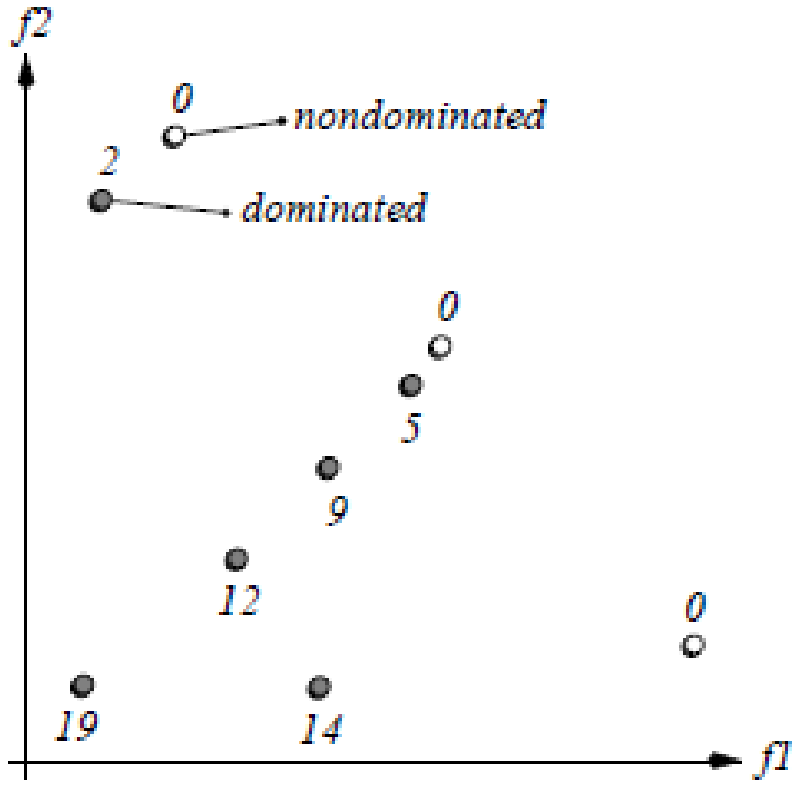


Illustration der raw fitness für zwei Funktionen $f1$ und $f2$, die zu maximieren sind; Quelle: [ZLT01]

Eigenschaften:

- Fitness ist zu minimieren; ein Element i wird genau dann von keinem anderen dominiert, wenn $R(i) = 0$ bzw. $F(i) < 1$.
- Komplexität:
  - $O((\mu + \lambda)^2)$ für Bestimmung von $R(i)$
  - $O((\mu + \lambda)^2 \log(\mu + \lambda))$ für Bestimmung von $D(i)$

Survivor Selection in SPEA2

Grundidee: Übernehme alle nicht-dominierten Lösungen (das größtmögliche approximation set) in die nächste Generation. Auffüllen nach Fitness, falls die Menge $< \mu$ ist; sonst Löschen von Elementen, in deren Umgebung sich viele weitere befinden. Formal:

1. $P_{t+1} = \{i \in P_t \cup Q_t \mid F(i) < 1\}$;
2. falls $|P_{t+1}| < \mu$, fülle in Reihenfolge der Fitness auf bis $|P_{t+1}| = \mu$;
3. falls $|P_{t+1}| > \mu$, lösche iterativ jeweils das Element i, für das $i \leq_{\mathrm{d}} j$ für alle $j \in P_{t+1}$. Definition $\leq_{\mathrm{d}}$:

$$i \leq_{\mathrm{d}} j \Leftrightarrow \forall 1 \leq k < |P_{t+1}| : \sigma_i^k = \sigma_j^k \vee$$
$$\exists 1 \leq k < |P_{t+1}| : \left[\left(\forall 1 \leq l < k : \sigma_i^l = \sigma_j^l\right) \wedge \sigma_i^k < \sigma_j^k\right]$$

Hierbei ist wieder σ_i^k der Abstand des Elements i zu seinem k-ten Nachbarn im objective space.

Illustration: Reduktion um Elemente

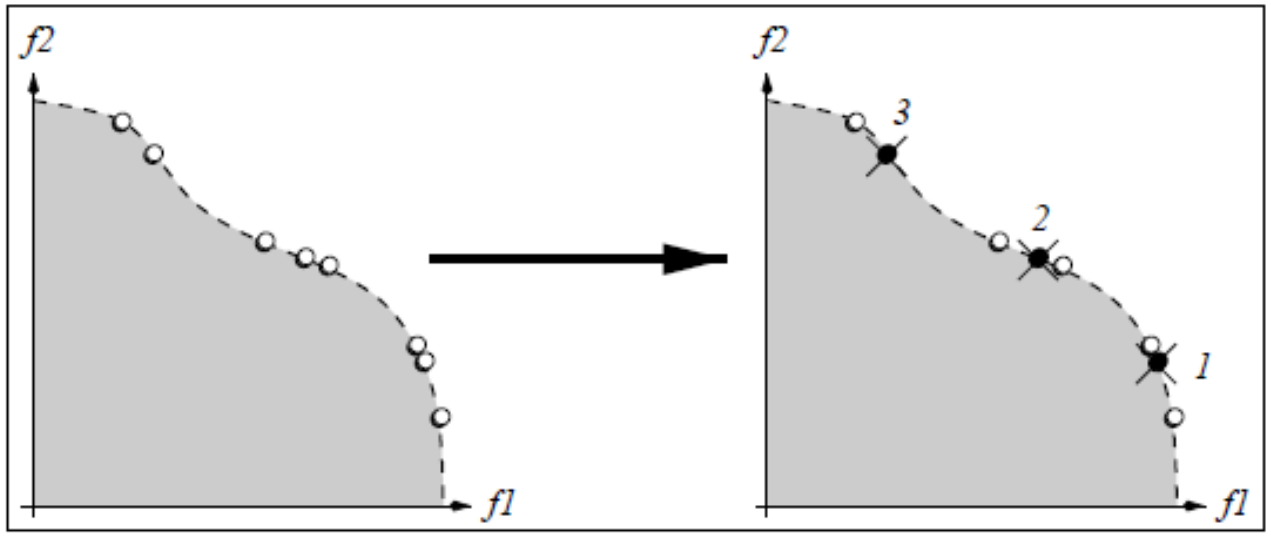


Illustration der Reduktion um 3 Elemente (in der Reihenfolge 1,2,3); Quelle: [ZLT01].

Eigenschaften:

- Reduktion der Menge soll so geschehen, dass die Form der approximation front möglichst beibehalten wird.
- Die Reduktion ist so definiert, dass die Randelemente nie gelöscht werden können.

Literatur

- [DAPM02] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, T. Meyarivan: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2): 182–197 (2002)
- [ZLT01] Eckart Zitzler, Marco Laumanns, Lothar Thiele: SPEA2: Improving the strength pareto evolutionary algorithm. TIK Report 103 (Mai 2001). https://doi.org/10.3929/ethz-a-004284029