UNIVERSITÄT DORTMUND

Wintersemester 2005/06

Fundamente der Computational Intelligence

(Vorlesung)

Prof. Dr. Günter Rudolph

Fachbereich Informatik

Lehrstuhl für Algorithm Engineering

Inhalt

- PBIL & Co.
- Differentialevolution
- Genetic Programming

PBIL: Population-Based Incremental Learning

- Baluha & Caruana (1995), Conference on Machine Learning
- Suchraum $\mathbb{B}^n$, Minimierung pseudo-boolescher Funktionen
- Population X mit μ Individuen

$$\begin{array}{ccl}
X = & \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{\mu 1} & x_{\mu 2} & \cdots & x_{\mu n} \end{pmatrix} & \begin{array}{l} \leftarrow \text{Individuum 1} \\ \leftarrow \text{Individuum 2} \\ \qquad \vdots \\ \leftarrow \text{Individuum } \mu \end{array} \\
& \begin{array}{cccc} \uparrow & \uparrow & & \uparrow \\ \text{Genpool 1} & \text{Genpool 2} & \ldots & \text{Genpool n} \end{array} &
\end{array}$$

$$\boxed{p_j = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{ij}}$$

Genpool-Rekombination:

Ziehe Gen j für Nachkomme Y gleichverteilt von Komponente j aller Individuen

$\Updownarrow$

Setze Gen j für Nachkomme Y auf 1 mit W‘keit $p_j = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{ij}$, sonst auf 0

$\Rightarrow$ Population charakterisiert durch Verteilungen p = (p_1, p_2, …, p_n)

Grundidee der Evolutionsschleife:

1. Generiere λ Nachkommen durch Genpool-Rekombination gemäß p
2. Selektiere μ beste Nachkommen als neue Eltern
3. Aktualisiere p gemäß Genpools der neuen Eltern

Seien $y_{1:\lambda}, y_{2:\lambda}, \ldots, y_{\lambda:\lambda}$ die λ sortierten Nachkommen mit

$$f(y_{1:\lambda}) \leq f(y_{2:\lambda}) \leq \cdots \leq f(y_{\lambda:\lambda})$$

Setze $p^{(0)} = (\frac{1}{2}, \ldots, \frac{1}{2})$ und $k = 0$

repeat

Erzeuge λ Nachkommen y_i gem $p^{(k)}$

Sortiere y_i gemäß ihrer Güte

$$p^{(k+1)} = (1-\alpha)\, p^{(k)} + \alpha \frac{1}{\mu} \sum_{k=1}^{\mu} y_{k:\lambda}^{(k)}$$

until Terminierungsregel erfüllt

$\alpha \in (0,1)$

Achtung!

Aktualisierungsregel $p^{(k+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} y_{i:\lambda}^{(k)}$ problematisch!
(α = 1)

$\Rightarrow$ kann zu jedem p $\in \mathbb{B}^n$ führen (durch unglückliches Ziehen)

Analyse:

- Kvasnicka et al. (1995): 1-dimensional, deterministisch
- Rudolph & Höhfeld (1997): Spezialfall μ = 1
- Gonzalez et al. (2001): 2-dimensional, leider falsch!
- ???

Analyse (Höhfeld & Rudolph 1997)

Aktualisierung: $p^{(k+1)} = (1-\alpha)\, p^{(k)} + \alpha\, b^{(k)}$

wobei $b^{(k)} = y^{(k)}_{1:\lambda}, \ \alpha \in (0,1), \ p^{(0)}_i = \frac{1}{2}.$

Ziel:

Wir suchen Grenzwert $p^{(\infty)}$ der stochastischen Folge $(p^{(k)} : k \geq 0)$!

Vorüberlegung:

p ist beschränkt, also Konvergenz in W'keit = Konvergenz im Mittel

also: $\lim\limits_{k \to \infty} E[\,p^{(k)}\,] = x \in \{0,1\}^n$ **?**

$$\mathsf{E}[\, p^{(t+1)} \,|\, p^{(t)} \,] = (1-\alpha)\, p^{(t)} + \alpha\, \underbrace{\mathsf{E}[\, b^{(t)} \,|\, p^{(t)} \,]}_{?}$$

$$\mathsf{E}[\, b \,|\, p \,] = \sum_{x \in \{0,1\}^n} x \cdot \underbrace{\mathsf{P}\{\, b = x \,\}}_{?}$$

$$\mathsf{P}\{\, b = x \,\} \; =$$

$$\mathsf{P}\{\, s = x \,\} \sum_{i=0}^{\lambda-1} \mathsf{P}\{\, f(s) > f(x) \,\}^{i} \cdot \mathsf{P}\{\, f(s) \geq f(x) \,\}^{\lambda-1-i}$$

$$\mathsf{P}\{\, s = x \,\} \;=\; \prod_{i=1}^{n} p_i^{x_i}\,(1-p_i)^{1-x_i}$$

$$\mathsf{P}\{\, f(s) = f(x) \,\} \;=\; \sum_{\substack{y\in\{0,1\}^n \\ f(y)=f(x)}} \mathsf{P}\{\, s = y \,\}$$

$$\mathsf{P}\{\, f(s) > f(x) \,\} \;=\; \sum_{\substack{y\in\{0,1\}^n \\ f(y)>f(x)}} \mathsf{P}\{\, s = y \,\}$$

Beispiel: $n = 2$, $\lambda = 2$, $f(x) = \sum_i x_i \to$ min! (counting ones)

$$\mathsf{P}\left\{ b = \binom{0}{0} \right\} = 1 - (p_1 + p_2 - p_1 p_2)^2$$

$$\mathsf{P}\left\{ b = \binom{0}{1} \right\} = (1 - p_1)\, p_2\, (p_1 + p_2)$$

$$\mathsf{P}\left\{ b = \binom{1}{0} \right\} = p_1\, (1 - p_2)\, (p_1 + p_2)$$

$$\mathsf{P}\left\{ b = \binom{1}{1} \right\} = (p_1\, p_2)^2$$

$$\mathsf{E}[\, b \,|\, p \,] := F_2(p) = \begin{pmatrix} p_1\, (1 - p_2)\, (p_1 + p_2) + (p_1\, p_2)^2 \\ p_2\, (1 - p_1)\, (p_1 + p_2) + (p_1\, p_2)^2 \end{pmatrix}$$

Vektorfeld der Abbildung $F_2(p)$

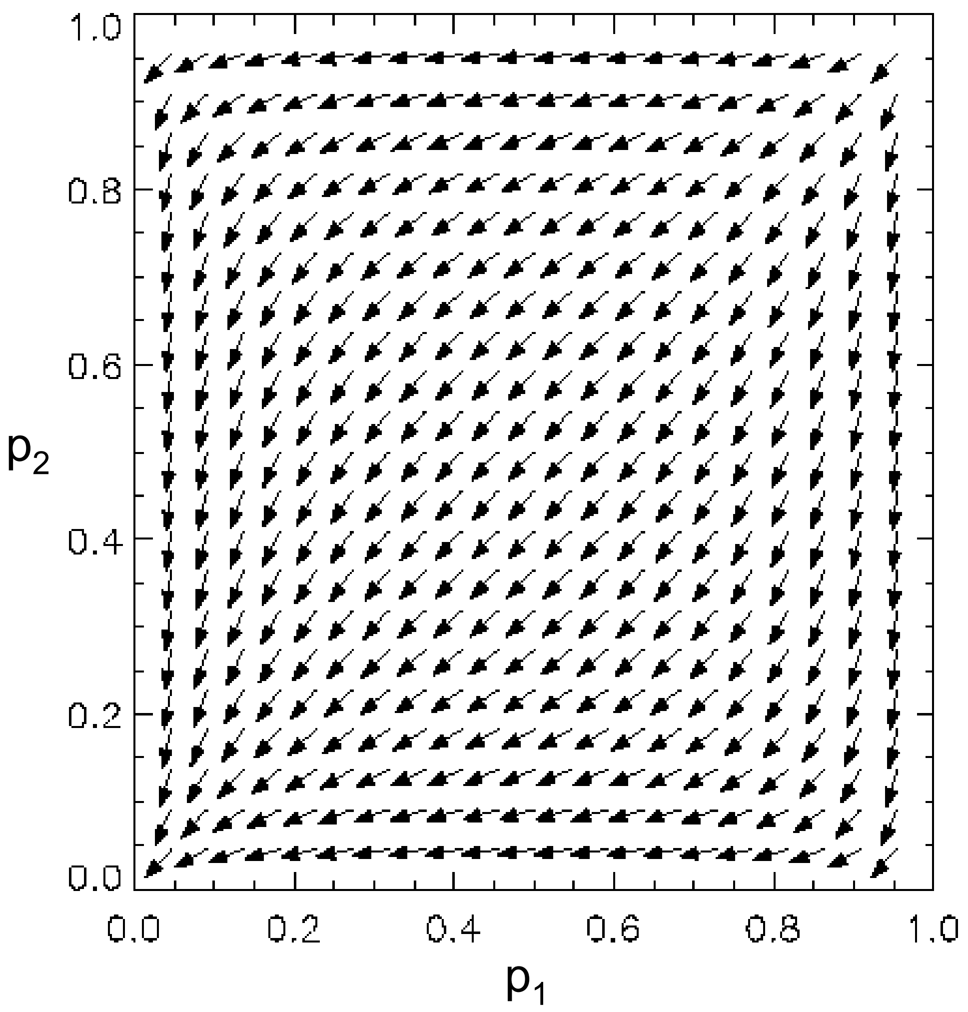
augenscheinlich: $F_2(p) < p$

Interpretation:

Unabhängig vom Startwert $p^{(0)}$ und unabhängig von aktueller Position $p^{(k)}$ hat $E[p^{(k+1)}]$ eine Tendenz in Richtung Optimum.

bzw.

$$E[p^{(k)}] \xrightarrow{\text{i.M.}} x^* \text{ für } k \to \infty$$



Satz:

Sei $f(x) = c' x$ eine lineare pseudo-boolesche Funktion, die zu minimieren ist.

Für $c_i < 0$ gilt $E[\, b_i \mid p \,] > p_i$, für $c_i > 0$ gilt $E[\, b_i \mid p \,] < p_i$, wobei b das selektierte beste Individuum ist.

Daraus folgt: PBIL konvergiert im Mittel zum Optimum linearer Funktionen.

Beweis: (Rudolph & Höhfeld 1997, S. 3f) ■

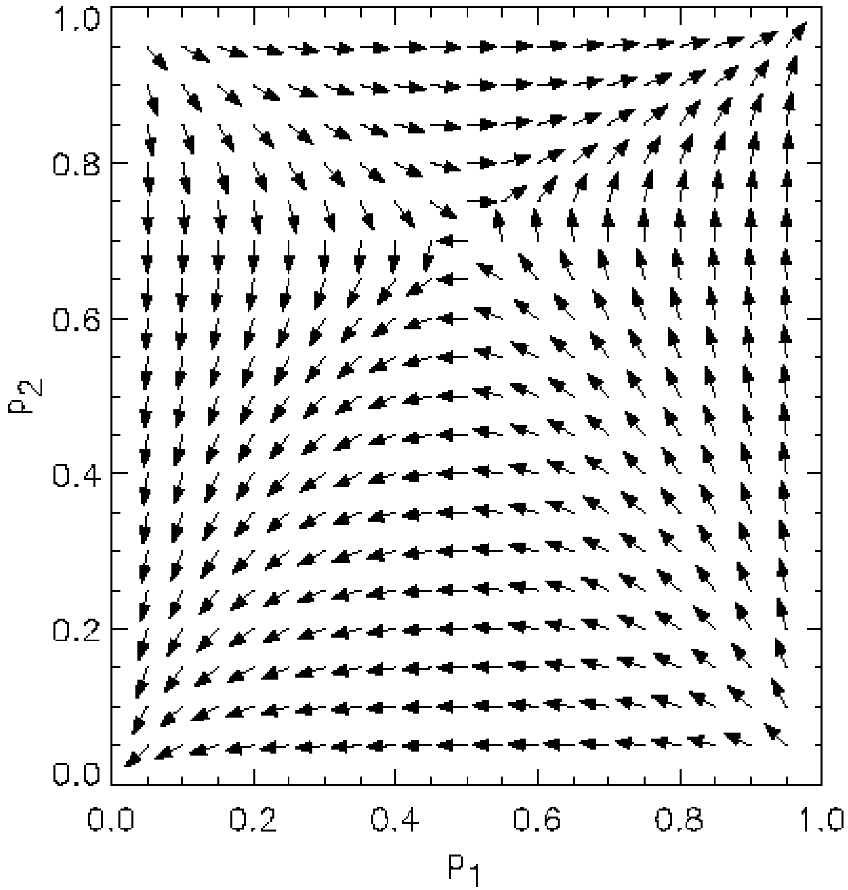
→ Lineare Funktionen sind einfach! (1+1)-EA braucht i.M. O(n log n) Schritte.

→ Ähnliches Resultat für nichtlineare Funktionen?

→ Wenn Beweise nicht gelingen wollen, dann Gegenbeispiel finden!

$f(00) < f(11) < f(01) < f(10)$

z. B. $f(x) = 3x_1 + 2x_2 - 4x_1x_2$



Interpretation:

Abhängig vom Startwert $p^{(0)}$ Tendenz zum lokalen oder globalen Optimum!

Kein globales Verfahren!

- Price (1996) + Storn (1996)
- ($\mu + \mu$)-EA mit speziellem Variationsoperator

initialisiere μ Individuen P = (X_1, X_2, …, X_μ)

```
repeat
  Q = ( )                          // leere Liste
  foreach  Individuum x ∈ P:
    wähle 3 Individuen ≠ x aus P ohne Zurücklegen  // ergibt (a,b,c)
    y = Variation(x, a, b, c)
    if  f(y) < f(x)  then Q.add(y) else Q.add(x)
  endfor
  P = Q
until  Terminierung
```

```
Variation(x, a, b, c) =

  wähle gleichverteilt ein j aus {1, 2, …, n}

  for i = 1 to n

    if i = j or mit W'keit p:
      y_i = a_i + α(b_i – c_i)
    else
      y_i = x_i

  endfor

  return y
```

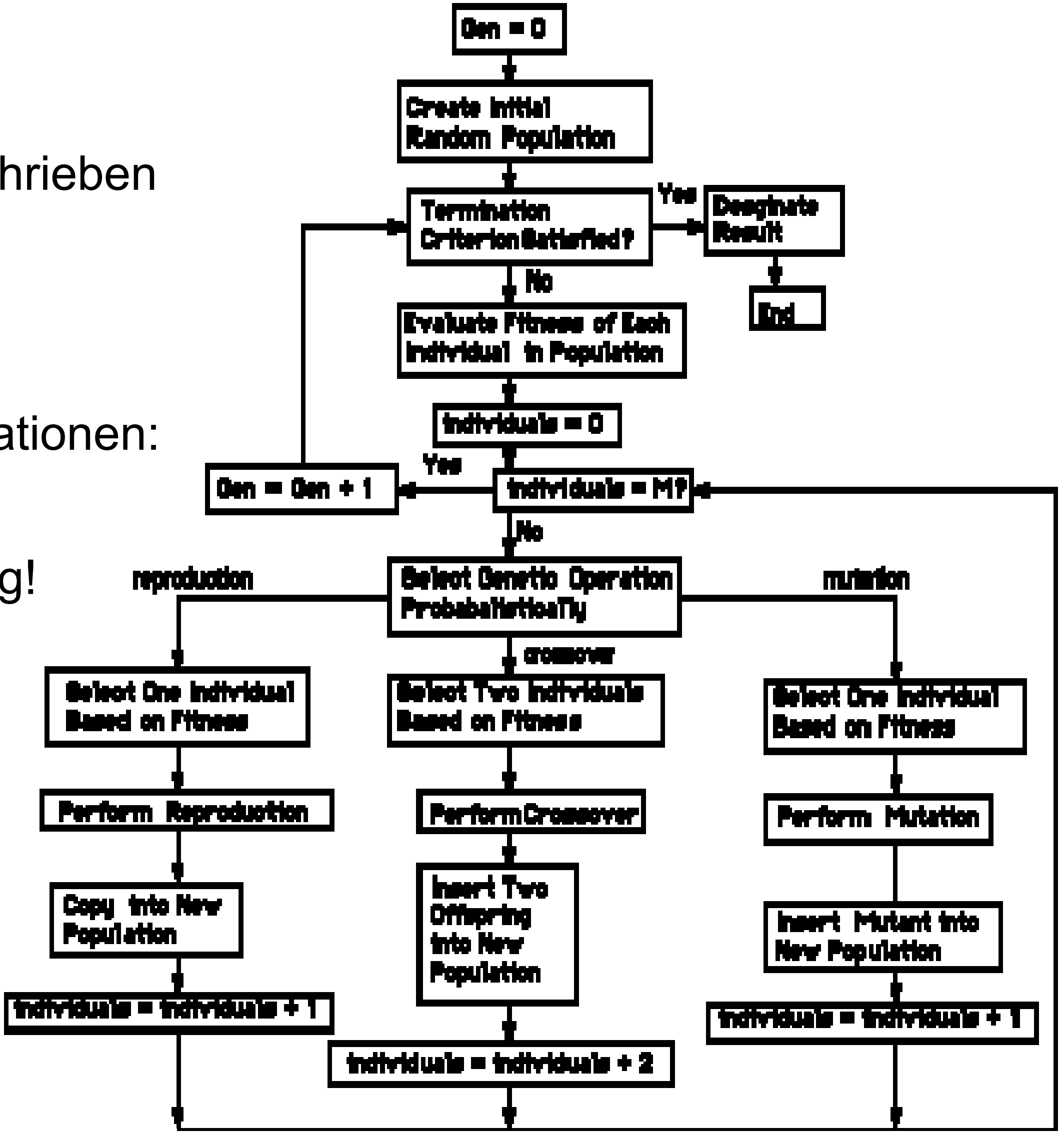
p: Rekombinationsrate

$\alpha \in (0,1)$

Problem:

Der kleinste Hyperquader, der die initiale Population beinhaltet, kann nicht verlassen werden!

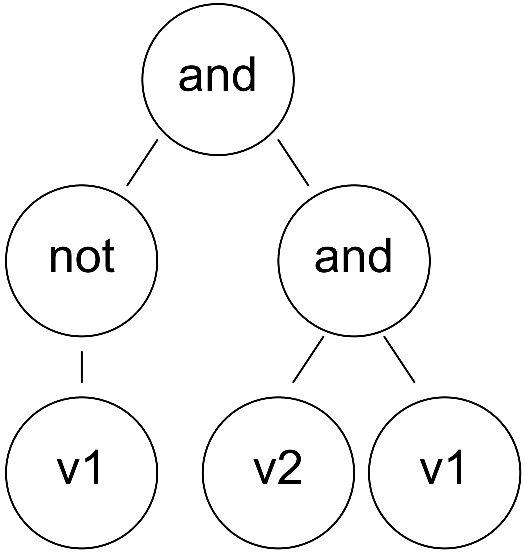
- entstanden im GA-Umfeld
- wird meist Koza (1989) zugeschrieben
- Evolution von Programmen
- Suchraum S: Syntaxbäume
- später auch andere Repräsentationen: z.B. Assembler, binary GP, …
- neue Variationsoperatoren nötig!

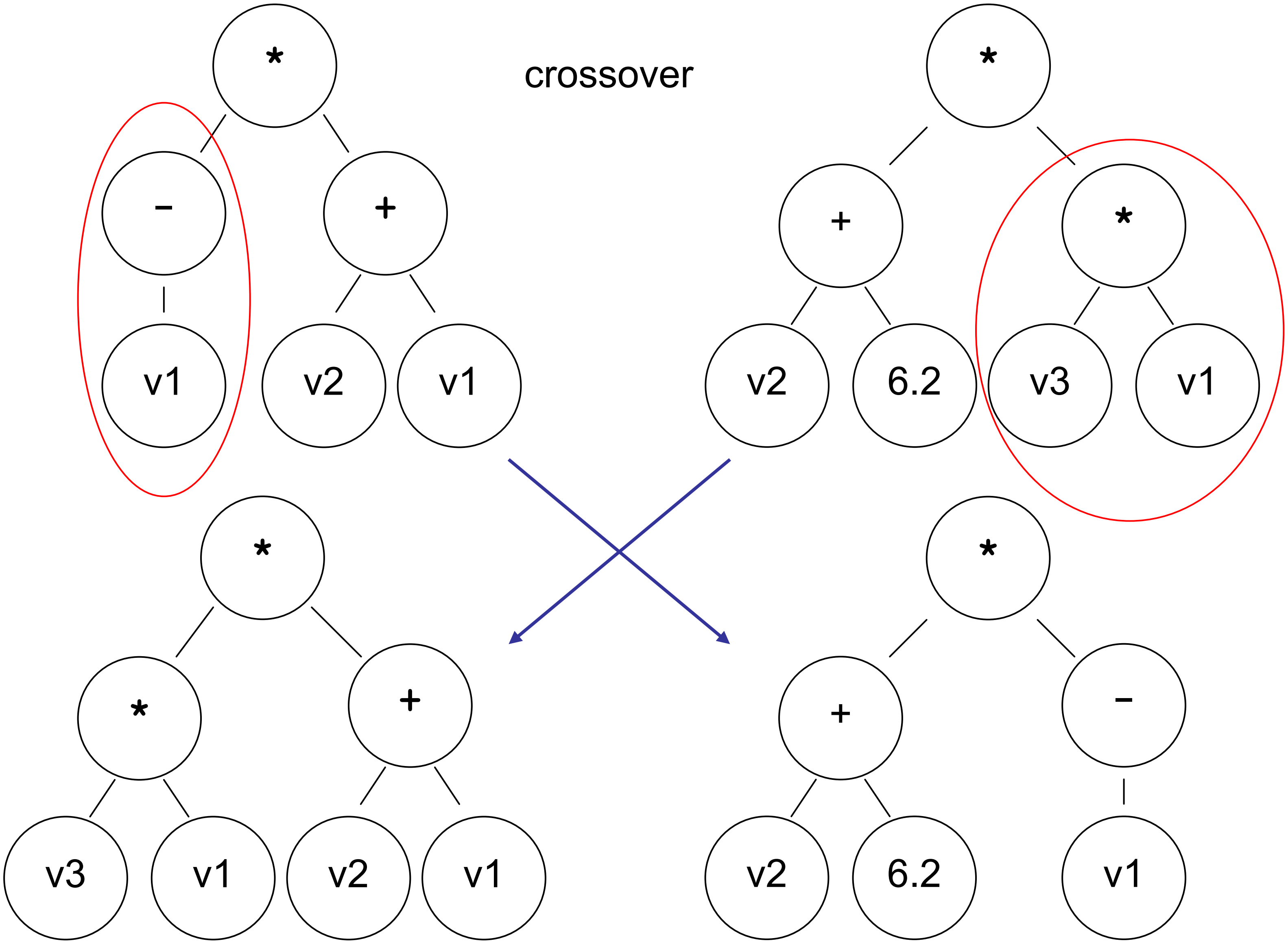


Klasse `tree`

Methoden:

```
createTree()
delete(node)
getSubtree(node)
insertTreeAt(node, tree)
contains(node)
size()
...
```




crossover
*
–
+
v1
v2
v1
*
+
*
v2
6.2
v3
v1
*
*
+
v3
v1
v2
v1
*
+
–
v2
6.2
v1

Mutation:

- Ersetzen eines Teilbaums durch zufälligen Baum
- Nur Konstanten werden verändert (keine Operationen)
- Wird eher selten eingesetzt
  → meistens riesige Populationen nur mit Crossover

Typische Anwendung

Symbolische Regression (Evolvieren einer Formel)

Theorie

sehr junges Feld (> 2000), Poli et al.
Schema-Theorie genannt, ist aber i.W. Markoff-Theorie