Genetic Algorithms in the Real World
Part III: The NSGA-II Algorithm

CSCI 315: Artificial Intelligence
Simon D. Levy
Fall 2007
Setting the Stage for NSGA-II

• Tuning parameters (for fitness sharing) are hard to work with.

• Lack of elitism (keeping best-ranked members) can lead to sub-optimal solutions.

• (Naive) nondominated sorting is $O(M N^3)$, where $N$ is population size.
Fitness Sharing without Tuning Parameters

- Crowding Distance \( d_i = \frac{(w + h)}{2} \)
- \( d_0 = d_1 = \infty \)
Crowding-Distance Assignment

crowding-distance-assignment(\mathcal{I})

\begin{align*}
\bar{l} &= |\mathcal{I}| \\
\text{for each } i, \text{ set } \mathcal{I}[i]_{\text{distance}} &= 0 \\
\text{for each objective } m \\
\mathcal{I} &= \text{sort}(\mathcal{I}, m) \\
\mathcal{I}[1]_{\text{distance}} &= \mathcal{I}[\bar{l}]_{\text{distance}} = \infty \\
\text{for } i = 2 \text{ to } (\bar{l} - 1) \\
\mathcal{I}[i]_{\text{distance}} &= \mathcal{I}[i]_{\text{distance}} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)/(f^m_{\text{max}} - f^m_{\text{min}})
\end{align*}

number of solutions in \mathcal{I}
initialize distance

sort using each objective value
so that boundary points are always selected
for all other points
Using Crowding for Selection

- Favor rank over crowding as selection criterion
- If ranks are same, favor less crowded solutions
- Define a partial order $<_n$ on solutions:
  
  $i <_n j$ if $(\text{rank}_i < \text{rank}_j)$
  
  or $((\text{rank}_i = \text{rank}_j)$
  
  and $(d_i > d_j)$
Elitism through “Incest”

• Problem with generational approach: no guarantee that children are fitter than parents

• So, given parent population $P_t$ and child population $Q_t$, create “super-population” $R_t = P_t \cup Q_t$ ($t =$ time, generation)

• Then sort $R_t$ according to $<_n$ to get $P_{t+1}$
Graphically

Fig. 2. NSGA-II procedure.
Reducing $O(M N^3)$ Sorting to $O(M N^2)$: Fast Nondominated Sort

- First step: for each solution $p$, compute
  - Domination count $n_p = \# \text{ of solutions that dominate } p$
  - Set of solutions $S_p$ that $p$ dominates
  - This computation is $O(M N)$ for each solution $p$, so for $N$ such solutions, it’s $O(M N^2)$
Fast Nondominated Sort

• Next: for each solution $p$ with $n_p = 0$, visit each member $q$ of its set $S_p$ and reduce $q$’s domination count by one.

• In doing so, if for any $q$ the domination count $n_q$ becomes zero, we put $q$ in a separate set $Q$, the second front

• Repeat above two steps with $Q$, until all fronts are identified
## Example

<table>
<thead>
<tr>
<th>$p$</th>
<th>$n_p$</th>
<th>$S_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>G, H</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>G, H</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>G, H</td>
</tr>
<tr>
<td>G</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
# Example

<table>
<thead>
<tr>
<th>$p$</th>
<th>$n_p$</th>
<th>$S_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>G, H</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>G, H</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>G, H</td>
</tr>
<tr>
<td>G</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th>$p$</th>
<th>$n_p$</th>
<th>$S_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>G, H</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>G, H</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>G, H</td>
</tr>
<tr>
<td>G</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>$p$</th>
<th>$n_p$</th>
<th>$S_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>G, H</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>G, H</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>G, H</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$p$</td>
<td>$n_p$</td>
<td>$S_p$</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>D, E, F, G, H</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>G, H</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>G, H</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>G, H</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Second Front
Fast Nondominated Sort: Complexity

- First step is $O(M N^2)$
- Second step $O(N^2)$
- So overall algorithm is $O(M N^2)$
Fast Nondominated Sort

\[
\text{fast-non-dominated-sort}(P)
\]

for each \( p \in P \)

\[
S_p = \emptyset \quad n_p = 0
\]

for each \( q \in P \)

\[
\begin{align*}
\text{if } (p \prec q) & \text{ then } \\
S_p &= S_p \cup \{q\} \quad n_p = n_p + 1
\end{align*}
\]

\[
\text{else if } (q \prec p) & \text{ then }
\]

\[
S_p = S_p \quad n_p = n_p
\]

\[
\text{if } n_p = 0 & \text{ then }
\]

\[
F_{\text{rank}} = 1 \quad F_1 = F_1 \cup \{p\}
\]

\[i = 1\]

\[Q = \emptyset\]

while \( F_i \neq \emptyset \)

\[
Q = \emptyset
\]

for each \( p \in F_i \)

\[
\text{for each } q \in S_p
\]

\[
\begin{align*}
q_{\text{rank}} &= i + 1 \\
Q &= Q \cup \{q\}
\end{align*}
\]

\[i = i + 1\]

\[F_i = Q\]

If \( p \) dominates \( q \)

Add \( q \) to the set of solutions dominated by \( p \)

Increment the domination counter of \( p \)

\( p \) belongs to the first front

Initialize the front counter

Used to store the members of the next front

\( q \) belongs to the next front
Altogether: NSGA-II
(One Generation)

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

combine parent and offspring population
\[ \mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots), \text{ all nondominated fronts of } R_t \]

until the parent population is filled
calculate crowding-distance in \( \mathcal{F}_i \)
include \( i \)th nondominated front in the parent pop
check the next front for inclusion
sort in descending order using \( \prec_n \)
choose the first \( (N - |P_{t+1}|) \) elements of \( \mathcal{F}_i \)
use selection, crossover and mutation to create
a new population \( Q_{t+1} \)
increment the generation counter