

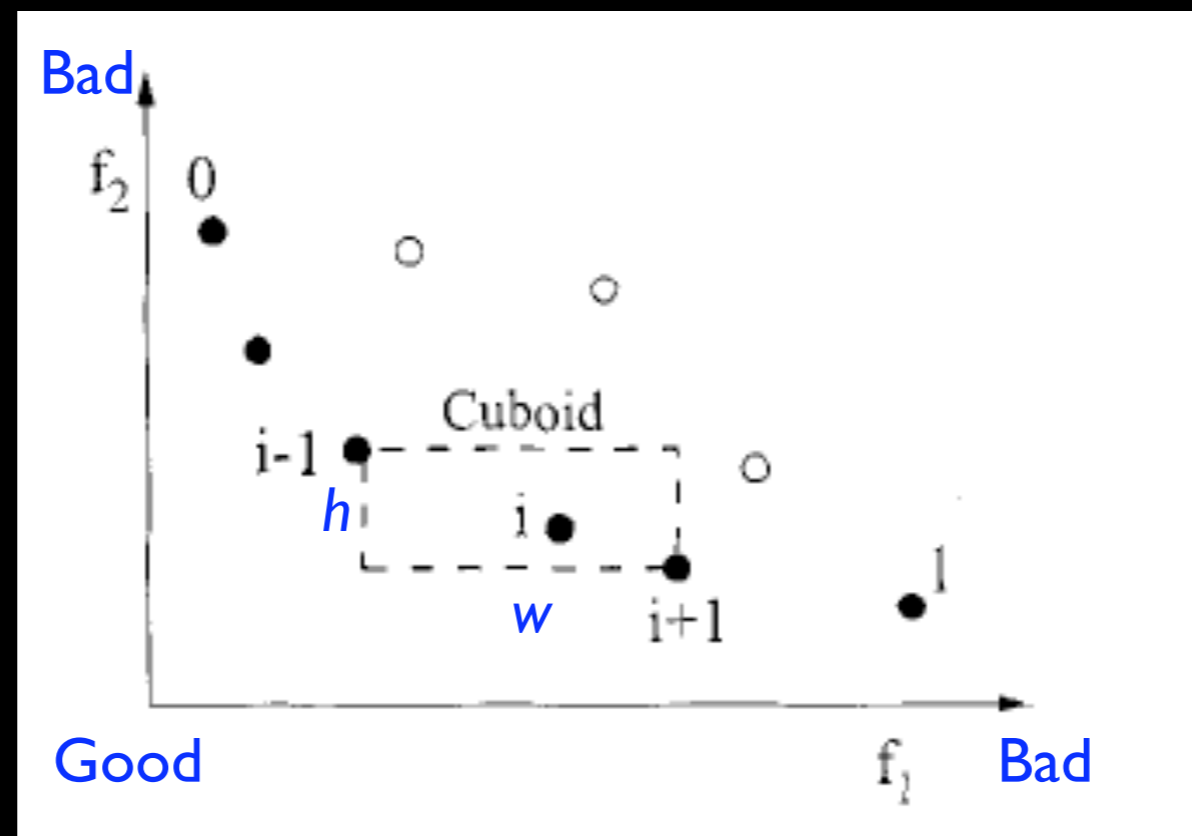
Genetic Algorithms in the Real World Part III: The NSGA-II Algorithm

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** $d_i = (w + h) / 2$
- $d_0 = d_1 = \infty$

Crowding-Distance Assignment

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

$l = |\mathcal{I}|$

for each i , set $\mathcal{I}[i]_{\text{distance}} = 0$

for each objective m

$\mathcal{I} = \text{sort}(\mathcal{I}, m)$

$\mathcal{I}[1]_{\text{distance}} = \mathcal{I}[l]_{\text{distance}} = \infty$

for $i = 2$ to $(l - 1)$

$\mathcal{I}[i]_{\text{distance}} = \mathcal{I}[i]_{\text{distance}} + (\mathcal{I}[i + 1].m - \mathcal{I}[i - 1].m) / (f_m^{\max} - f_m^{\min})$

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** \prec_n on solutions:
 - $i \prec_n j$ 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

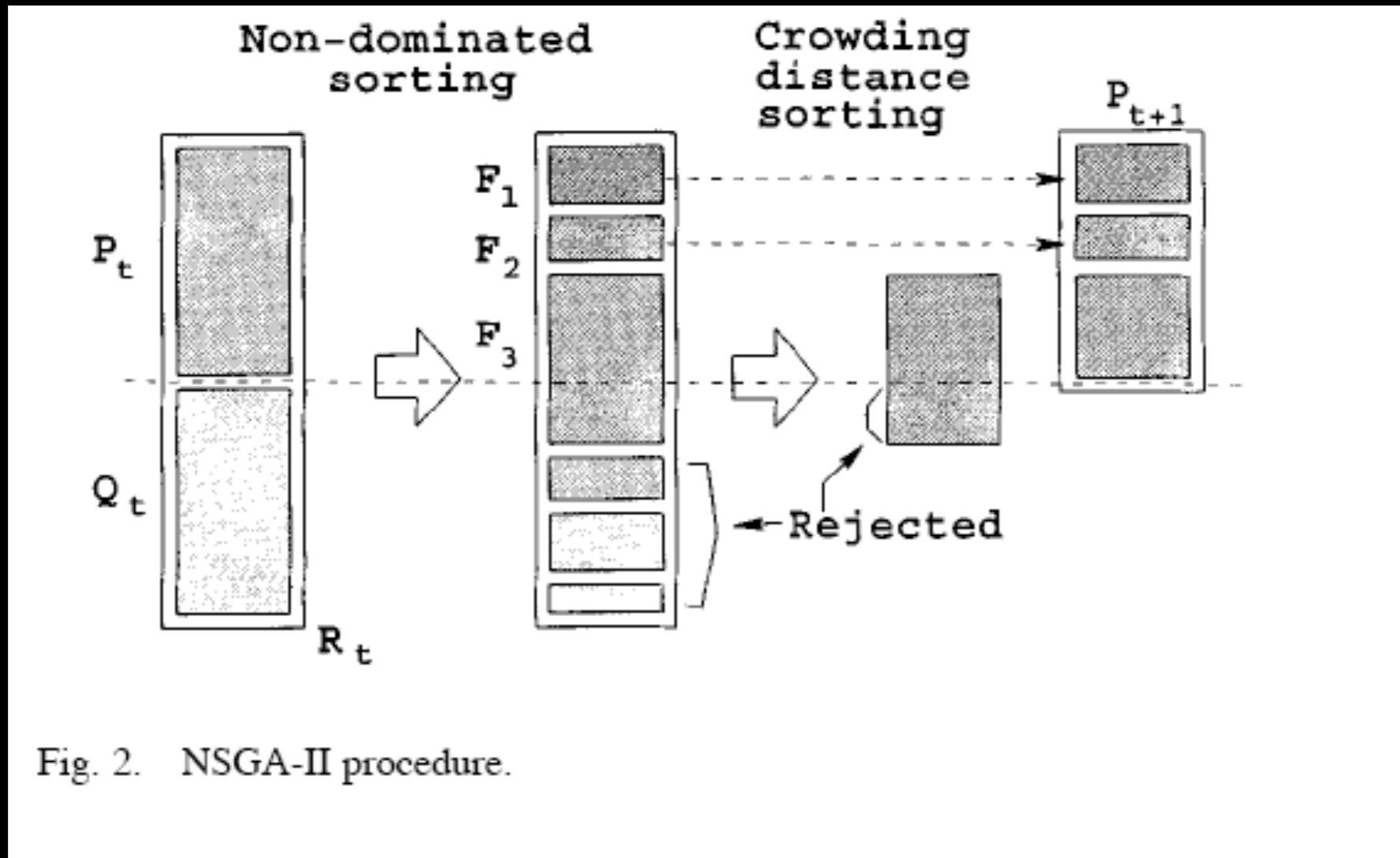


Fig. 2. NSGA-II procedure.

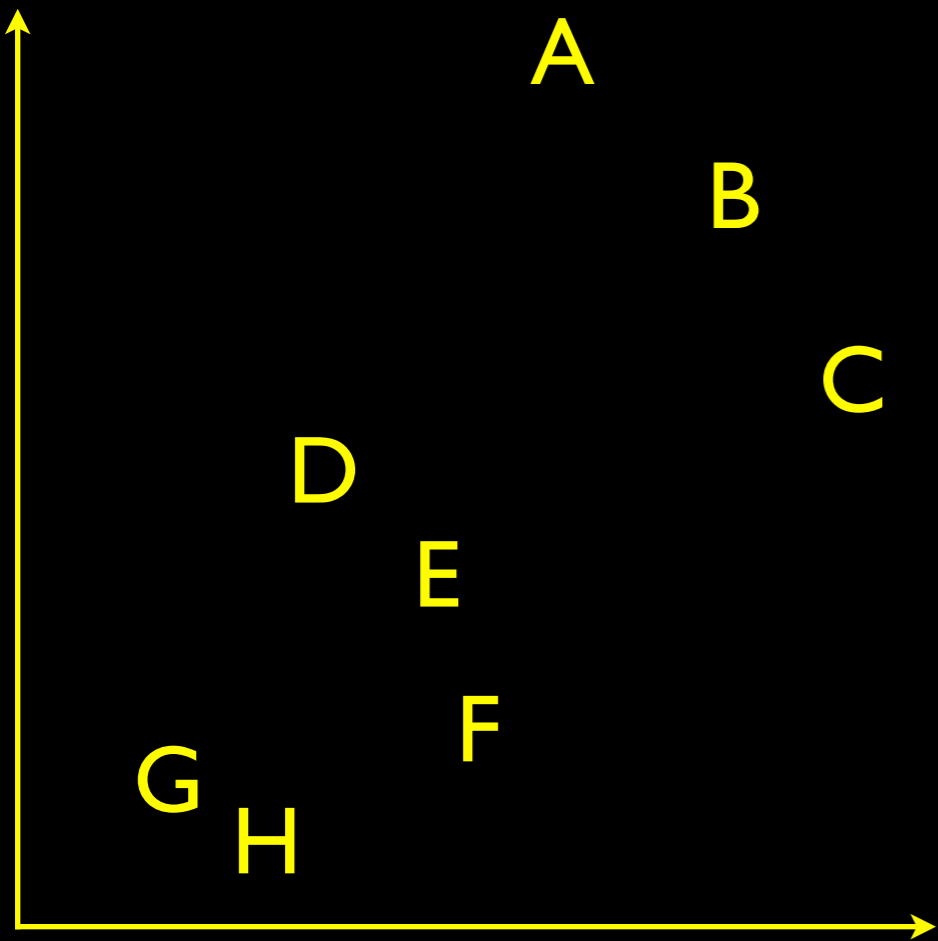
Reducing $O(M N^3)$ Sorting to $O(M N^2)$: Fast Nondominated Sort

- First step: for each solution p , compute
 - Domination count $n_p = \#$ 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



p	n_p	S_p
A	0	D, E, F, G, H
B	0	D, E, F, G, H
C	0	D, E, F, G, H
D	3	G, H
E	3	G, H
F	3	G, H
G	6	
H	6	

Example

p	n_p	S_p
A	0	D, E, F, G, H
B	0	D, E, F, G, H
C	0	D, E, F, G, H
D	2	G, H
E	2	G, H
F	2	G, H
G	5	
H	5	

Example

p	n_p	S_p
A	0	D, E, F, G, H
B	0	D, E, F, G, H
C	0	D, E, F, G, H
D	1	G, H
E	1	G, H
F	1	G, H
G	4	
H	4	

Example

p	n_p	S_p
A	0	D, E, F, G, H
B	0	D, E, F, G, H
C	0	D, E, F, G, H
D	0	G, H
E	0	G, H
F	0	G, H
G	3	
H	3	

Example

p	n_p	S_p
A	0	D, E, F, G, H
B	0	D, E, F, G, H
C	0	D, E, F, G, H
D	0	G, H
E	0	G, H
F	0	G, H
G	3	
H	3	

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

fast-non-dominated-sort(P)

for each $p \in P$

$S_p = \emptyset$

$n_p = 0$

for each $q \in P$

if $(p \prec q)$ then

$S_p = S_p \cup \{q\}$

else if $(q \prec p)$ then

$n_p = n_p + 1$

if $n_p = 0$ then

$p_{\text{rank}} = 1$

$\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$

$i = 1$

while $\mathcal{F}_i \neq \emptyset$

$Q = \emptyset$

for each $p \in \mathcal{F}_i$

for each $q \in S_p$

$n_q = n_q - 1$

if $n_q = 0$ then

$q_{\text{rank}} = i + 1$

$Q = Q \cup \{q\}$

$i = i + 1$

$\mathcal{F}_i = Q$

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)

$R_t = P_t \cup Q_t$

$\mathcal{F} = \text{fast-non-dominated-sort}(R_t)$

$P_{t+1} = \emptyset$ and $i = 1$

~~while~~ 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)

$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$

$Q_{t+1} = \text{make-new-pop}(P_{t+1})$

$t = t + 1$

combine parent and offspring population

$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$, 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