MULTIOBJECTIVE OPTIMISATION

Contents

1

Literature:

1. *Multi-Objective Optimisation using Evolutionary Algorithms*, Kalyanmoy Deb. Willey, 2004
2. *Evolutionary algorithms for solving multi-objective problems*, Carlos Coello Coello, Gary Lamont, David Van Veldhuizen, 2nd Edition, Springer, 2007.
3. A Comprehensive Survey of Evolutionary-based Multiobjective Optimisation Techniques, C. Coello Coello, *Knowledge and Information Systems*, Vol.1, No. 3, 1999, p.269-308
4. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art, D. Van Veldhuizen, G. Lamont, *Evolutionary Computation*, Vol.8, No. 2, 1999, p.125-147
5. Multi-objective Optimisation, K. Deb in: *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies*, E.K.Burke and G.Kendall (editors), Springer, 2005.

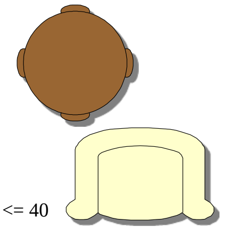
2

6. *Metaheuristics for Multiobjective Optimisation*, X. Gandibleux, M.Sevaux, K.Sorensen, V.T'kindt (editors) Lecture Notes in economics and Mathematical Systems, Vol. 535, Springer, 2004.
7. *Modern Heuristic Techniques for Combinatorial Problems*, (Ed) C.Reeves 1995, McGraw-Hill. Chapter 4.
8. Evolutionary Multiobjective Bibliography: *http://www.lania.mx/~ccoello/EMOO/*
9. PISA (**P**latform and programming language independent **I**nterface for **S**earch **A**lgorithms) website: *http://www.tik.ee.ethz.ch/pisa/*

3

From modelling and analysis lecture:

Linear programming



Variables
T = number of tables made per week
C = number of chairs made per week

Constraints

total work time	6T + 3C <= 40
customer demand	C >= 3T
storage space	C + 4T <= 16
	all variables >= 0

Objective
maximise 30T + 10C

1. Basic Concepts

Many real world optimisation problems can be formulated as non-linear programming problems with multiple objectives. e.g.:

minimise $f_1(x)$
minimise $f_2(x)$
maximise $f_3(x)$
subject to constraints $C(x) = (c_1(x),\ldots,c_m(x))$
such that $x=(x_1,\ldots,x_n)$, $x \in X$

where $\boldsymbol{X}$ is the decision space
x is the decision vector

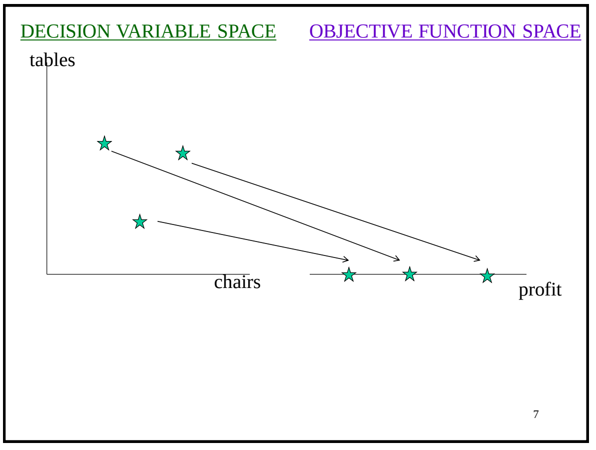
5

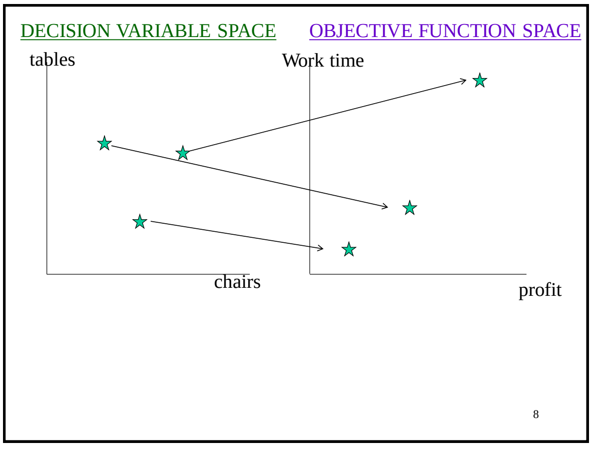
1. Basic Concepts

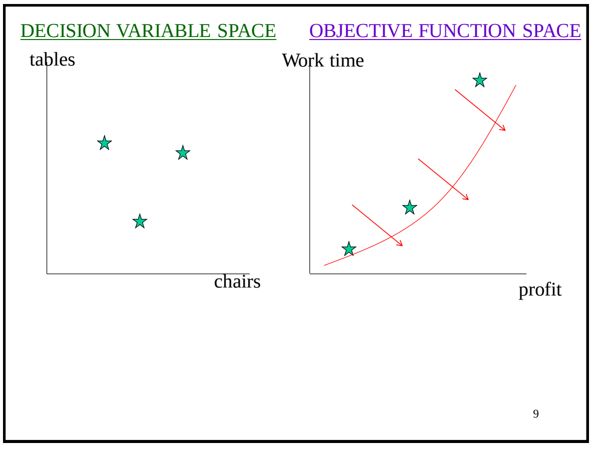
General problem statement
minimise $F(x) = (f_1(x),\ldots,f_k(x))$,
subject to constraints $C(x) = (c_1(x),\ldots,c_m(x))$
such that $x=(x_1,\ldots,x_n)$, $x \in X$

where $\boldsymbol{X}$ is the decision space
$\boldsymbol{Y}$ is the objective space
x is the decision vector
F: $\boldsymbol{X} \rightarrow \boldsymbol{Y}$

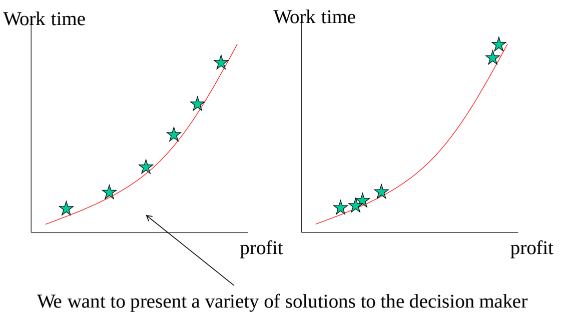
6


DECISION VARIABLE SPACE
OBJECTIVE FUNCTION SPACE
tables
chairs
profit
7


DECISION VARIABLE SPACE
OBJECTIVE FUNCTION SPACE
tables
Work time
chairs
profit
8

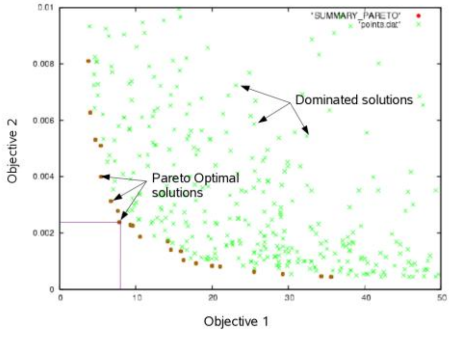

DECISION VARIABLE SPACE
OBJECTIVE FUNCTION SPACE
tables
Work time
chairs
profit
9

OBJECTIVE FUNCTION SPACE

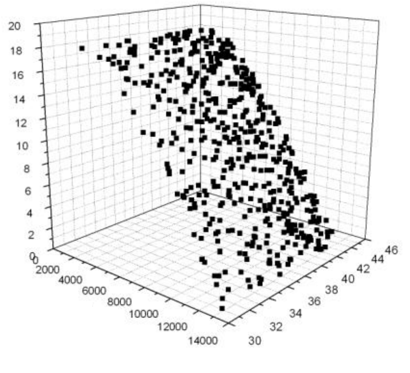


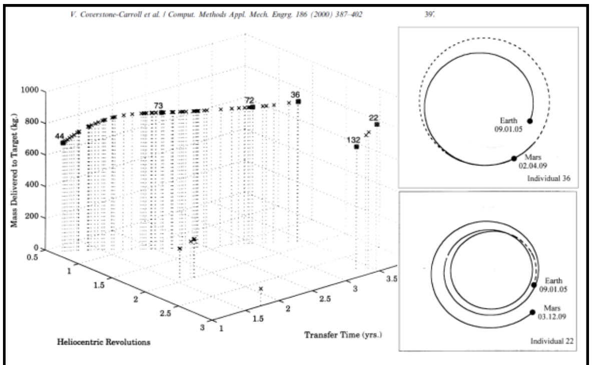
We want to present a variety of solutions to the decision maker

10



11





13

Dominance relation

Dominance relation, used by almost all MO algorithms

Dominance is denoted by u ◁ v (u dominates v)

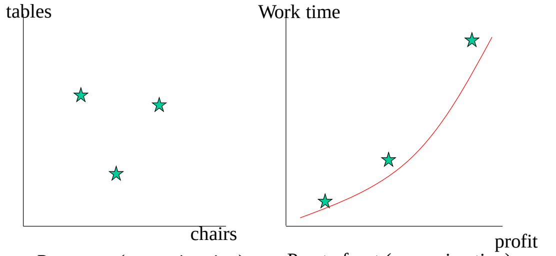
Among a set of solutions, the non-dominated set is where each dominates any solution outside the set.

14

- Pareto dominance
  An objective vector $u = (u_1,\ldots,u_k)$, is said to dominate another objective vector $v = (v_1,\ldots,v_k)$, (denoted by u ◁ v) iff $u_i \leq v_i$, $i=1,\ldots k$, and there exists $i \in \{1,\ldots, k\}$, $u_i < v_i$ (for a minimisation problem)
- Pareto optimality
  A solution $x \in X$, is said to be Pareto optimal with respect to X iff there is no $x' \in X$ for which $F(x')$ dominates F(x).
- The Pareto-optimal front contains those **objective vectors** that are not dominated by any other vector .
- The Pareto-optimal set contains those **decision vectors** whose corresponding objective vectors are not dominated by any other vector in the objective space.

15

DECISION VARIABLE SPACE OBJECTIVE FUNCTION SPACE



Pareto set (approximation)
non-dominated set of decision variable vectors

Pareto front (approximation)
non-dominated objective vectors

16

Dominance exercise

Identify the Pareto (non-dominated) front in each diagram

17

2. Aggregation-based Approaches

Weighted Sum Approach (Similar to last lecture)

$$\text{minimise} \quad \sum_{i=1}^{k} w_i f_i(x), \quad w_i \geq 0$$

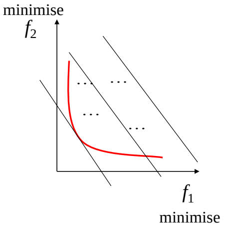
Simplifies the problem, as you now have one objective

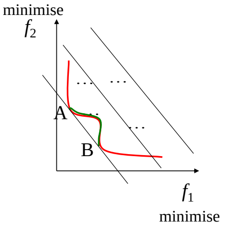
Drawbacks:

- Uniform spread of weights does not necessarily produce a uniform spread of points on the Pareto front.
- 'Non-convex' parts of the Pareto front cannot be obtained.

18

With a weighted sum approach, you cannot get to the points which are between A and B, even though they are Pareto optimal.

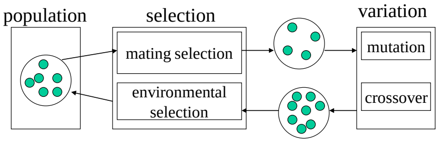




19

3. Evolutionary Multiobjective Optimisation

Evolutionary computation



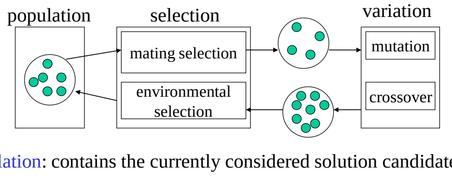
Maintains a population of solutions

As we want to find a set of solutions, intuitively evolutionary computation is a highly appropriate methodology

20

3. Evolutionary Multiobjective Optimisation

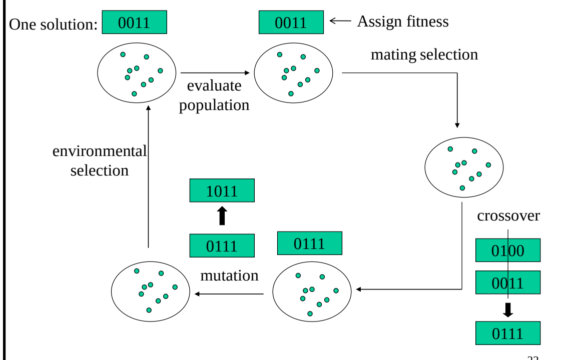
Evolutionary computation



Population: contains the currently considered solution candidates
Mating selection: takes (promising) solutions for variations
Variation: modifies solutions
 Mutation operator modifies individuals by changing small parts
 Crossover operator takes 2 ‘parents’ and combines them.
Environmental selection: determines which of the previously stored solutions and the newly created ones are kept in the population

21

Example. Four binary decision variables



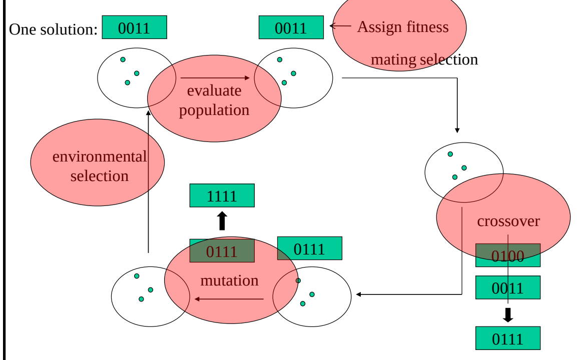
22

Pseudo-code of Evolutionary Algorithms (EAs)

Input: EAs parameters //*Population size, Operators' rates, Stopping Condition*
Output: a set of solutions
Generate the initial population Pop_0
$t = 0$
repeat
  for each individual $i \in Pop_t$ **do**
    Assign Fitness to individual i.
  end for
  Select solutions from Pop_t based on their fitness values
  Apply mutation/crossover operations on the selected solutions
  Perform environmental selection to obtain next generation Pop_{t+1}
  $t = t+1$
until the stopping condition is met.

23

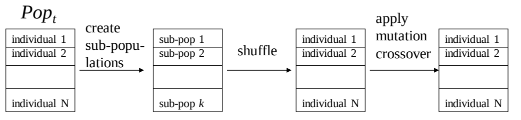
Example. Four binary decision variables



24

4. Non-pareto Based Evolutionary Algorithm

VEGA: The Vector Evaluated Genetic Algorithm, (Shaffer 1985)



Split the population, one subpopulation per objective. Within each subpopulation, the fitness is assigned based on its one objective.

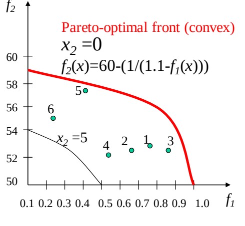
The main drawback:

- Selects individuals which are good at one objective, but leaves out compromise solutions.
- Non-convex parts of the Pareto front cannot be obtained.

25

Example.

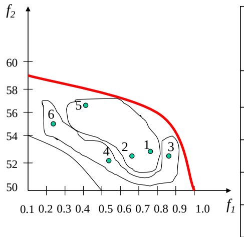
Maximise $f_1(x) = 1.1 - x_1$
Maximise $f_2(x) = 60 - (1+x_2)/x_1$
subject to $0.1 \le x_1 \le 1$,
$0 \le x_2 \le 5$
f_2 reaches maximum for $x_2 = 0$



Solution	x_1	x_2	f_1	f_2
1	0.31	0.89	0.79	53.90
2	0.43	1.92	0.67	53.21
3	0.22	0.56	0.88	52.91
4	0.59	3.63	0.51	52.15
5	0.66	1.41	0.44	56.35
6	0.83	2.51	0.27	55.77

26

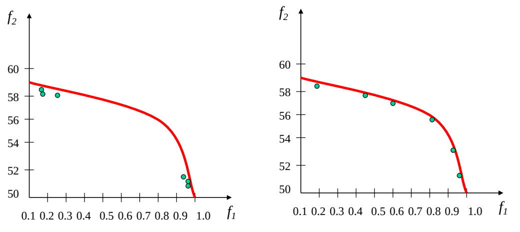
Randomly sort, and assign fitnesses



S.	x_1	x_2	f_1	f_2	Sub-pop	F
1	0.31	0.89	0.79	53.90	1	0.79
2	0.43	1.92	0.67	53.21	1	0.67
5	0.66	1.41	0.44	56.35	1	0.44
4	0.59	3.63	0.51	52.15	2	52.15
3	0.22	0.56	0.88	52.91	2	52.91
6	0.83	2.51	0.27	55.77	2	55.77

Selection: roulette wheel (fitness proportionate)
likely to have two copies of sol. 1 and one copy of sol. 2
+
likely to have two copies of sol. 6 and one copy of sol. 3

27



Drawback.
The population tends to converge to extreme solutions

28

6. Pareto-based Evolutionary Algorithm

Procedure for finding the nondominated set

Variables:

P - the Pareto front (approximation)
i - counter for solutions
j – counter for solutions from *P*
N – number of the solutions in the population

29

Step 1. Initialise P={1} // P (the nondominated set) contains the first solution
i=2

Step 2. j=1

Step 3. Compare solution i and solution j from P in terms of dominance
IF $i \lhd j$ // i dominates j
$P = P \setminus \{j\}$ //delete j from P
IF $j < |P|$ THEN $j = j + 1$
GO TO Step 3
ELSE GO TO Step 4
ELSE IF $j \lhd i$ // j dominates i
THEN GO TO Step 5.
ELSE IF $j < |P|$ THEN $j = j + 1$ // j and i do not dominate each other
GO TO Step 3

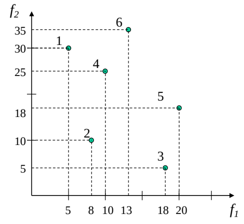
Step 4. $P = P \cup \{i\}$ // insert i in P

Step 5. IF $i < N$ // N is the total number of solutions
THEN $i = i + 1$
GO TO Step 2.
ELSE P is the approximation of Pareto front ; STOP

30

Example

Solution	f_1 (minimise)	f_2 (minimise)
1	5	30
2	8	10
3	18	5
4	10	25
5	20	18
6	13	35



31

Step 1. P={1}, i=2

Step 2. j=1

Step 3. compare *sol 2* and *sol 1*

sol 2 $\ntriangleleft$ *sol 1*

sol 1 $\ntriangleleft$ *sol 2*

$1 \not< 1 = |P|$

Step 4. $P = \{1\} \cup \{2\} = \{1, 2\}$

Step 5. $2 < 6 = |N|$

$i = 2+1 = 3$

GO TO Step 2

Step 2. j=1 // set P has *sol 1* and *sol 2*

Step 3. compare *sol* 3 and *sol 1*

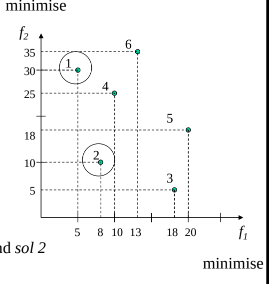
sol 3 $\ntriangleleft$ *sol 1*

sol 1 $\ntriangleleft$ *sol 3*

$1 < 2 = |P|$

j=1+1 = 2

GO TO Step 3



32

Step 3. compare *sol 3* and *sol 2*

sol 3 $\ntriangleleft$ *sol 2*

sol 2 $\ntriangleleft$ *sol 3*

$2 \not< 2 = |P|$

Step 4. $P = \{1, 2\} \cup \{3\} = \{1, 2, 3\}$

Step 5. $3 < 6 = |N|$

$i = 3+1 = 4$

GO TO Step 2

Step 2. j=1 // set P has *sol 1*, *sol2*, and *sol 3*

Step 3. compare *sol* 4 and *sol 1*

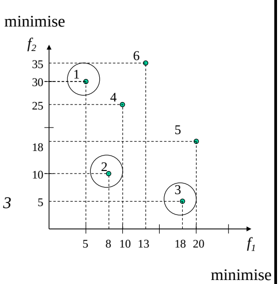
sol 4 $\ntriangleleft$ *sol 1*

sol 1 $\ntriangleleft$ *sol 4*

$1 < 3 = |P|$

j=1+1 = 2

GO TO Step 3



33

Step 3. compare *sol 4* and *sol 2*

sol 4 $\ntriangleleft$ *sol 2*

sol 2 $\triangleleft$ *sol 4*

Step 5. 4 < 6 = $|N|$

i = 4+1 =5

GO TO Step 2

Step 2. j=1 // set P has *sol 1, sol2,* and *sol 3*

Step 3. compare *sol* 5 and *sol 1*

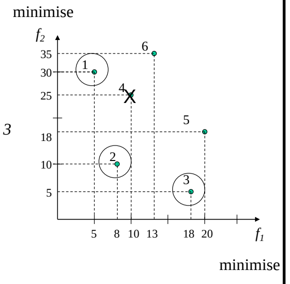
sol 5 $\ntriangleleft$ *sol 1*

sol 1 $\ntriangleleft$ *sol 5*

1< 3 = $|P|$

j=1+1 = 2

GO TO Step 3



34

Step 3. compare *sol* 5 and *sol 2*

sol 5 $\ntriangleleft$ *sol 2*

sol 2 $\triangleleft$ *sol 5*

Step 5. 5< 6 = $|N|$

i=5+1 = 6

GO TO Step 2

Step 2. j = 1

Step 3. compare *sol* 6 and *sol 1*

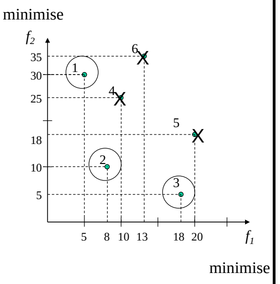
sol 6 $\ntriangleleft$ *sol 1*

sol 1 $\triangleleft$ *sol 6*

Step 5. $6 \not< 6$ = $|N|$

P = {1, 2, 3}

STOP



35

Non-dominated sorting procedure

Variables:

n_i – domination count: the number of solutions which dominate the solution i

S_i – set of solutions dominated by solution i

Pop – population

F_k – non-dominated front of level k

36

Step 1. For all $i \in Pop$

n_i=0 ; S_i=0

Step 2. For all $i \in Pop$

IF $i \lhd j$ // i dominates j

$S_i = S_i \cup \{j\}$

ELSE IF $j \lhd i$ // j dominates i

$n_i = n_i + 1$

Step 3. For all $i \in Pop$

IF n_i=0

keep i in the first non-dominated front F_1

37

---

k=1

Step 4. While $F_k \neq \varnothing$

Initialise $Q = \varnothing$ // Q stores next non-dominated solutions

For all $i \in F_k$

For all $j \in S_i$

$n_j = n_j - 1$

IF $n_j = 0$ THEN $Q = Q \cup \{j\}$

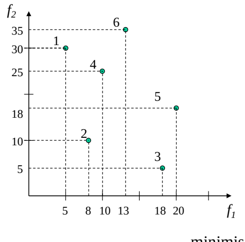
$k = k + 1$

$F_k = Q$

38

Example

minimise



minimise

i=1

1 $\lhd$ 6, S_1={6)

n_1=0

i=2

2 $\lhd$ 4, 2$\lhd$ 5, 2$\lhd$ 6, S_2={4, 5, 6}

n_2=0

i=3

3 $\lhd$ 5, S_3={5}

n_3=0

i=4

4 $\lhd$ 6, S_4={6}

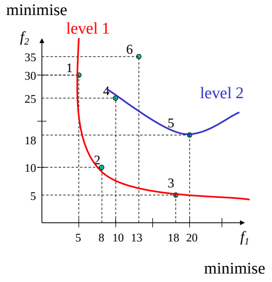
2 $\lhd$ 4, n_4=1

i=5

$S_5 = \varnothing$

2 $\lhd$ 5, 3$\lhd$ 5, n_5=2

39



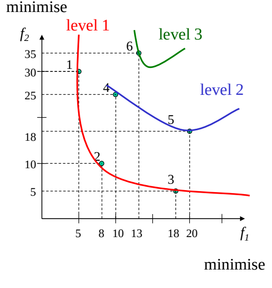
i=6
$S_6=\emptyset$
1 $\lhd$ 6, 2 $\lhd$ 6, 4 $\lhd$ 6, n_6=3

F_1 = {1, 2, 3}
k=1
1 $\in F_1$ S_1={6}
n_6=3-1=2

2$\in F_1$ S_2={4, 5, 6}
n_4=1-1=0, n_5=2-1=1, n_6=2-1=1

3$\in F_1$ S_3={5}
n_5=1-1=0
Q = {4, 5}

40



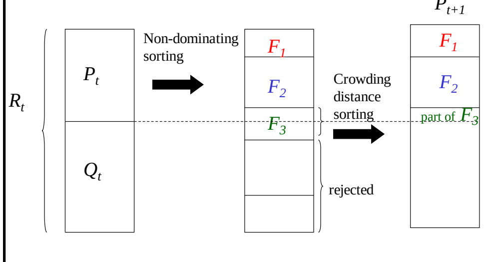
k=2
F_2 = {4, 5}
4 $\in F_2$ S_4={6}
n_6=1-1=0

5$\in F_2$ $S_5=\emptyset$
Q = {6}
F_3 = {6}

41

NSGA-II - Elitist Non-dominated Sorting GA

(Deb, Agrawal, Pratap, Meyarivan, 2002)



42

Variables: P_t parent population
Q_t offspring population
F_m different fronts, $m=1,\ldots,M$
N size of the population

Step 1. $R_t = P_t \cup Q_t$

Step 2. perform **Non-dominated sorting** to R_t
identify different fronts F_m, $m=1,\ldots,M$

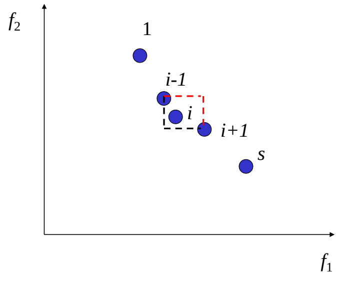
Step 3. $P_{t+1} = \varnothing$
$m= 1$
While $|P_{t+1}| + |F_m| < N$
$P_{t+1} = P_{t+1} \cup F_m$
$m = m+1$

Step 4. Perform **Crowding distance sorting**
Include the most widely spread (N-$|P_{t+1}|$) solutions in P_{t+1}

Step 5. Create offspring population Q_{t+1} from P_{t+1} by using the **crowded tournament selection**, crossover and mutation

43

<u>Crowding Distance Assignment Procedure</u>



44

Crowding-sort

Step 1. For all $i \in F_m$, $d_i=0$, $i=1,\ldots,s$

Step 2. For each objective k, $k=1,\ldots,K$
I_k=sort(f_k, >) // sorts the set from lowest to highest of f_k
// I_k is the list of indices

Step 3. For all $k=1,\ldots,K$
$d_{I_k 1} = d_{I_k s} = \infty$ // the distance for the extreme solutions set to infinity
// $I_{k\,j}$ is the index of the jth member in the sorted list I_k

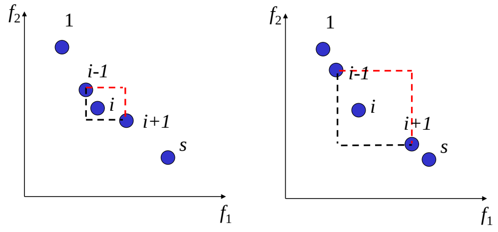
For all other solutions $j=2,\ldots,s-1$ ← Distance between the two solutions either side

$$d_{I_k j} = \frac{f_k^{(I_k\, j+1)} - f_k^{(I_k\, j-1)}}{f_k^{\max} - f_k^{\min}}$$ ← Normalise the distance

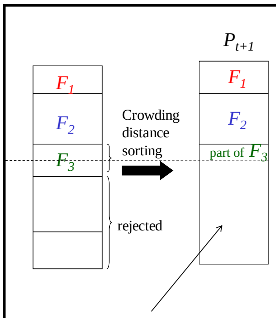
$d_j = d_j + d_{I_k j}$

45

A larger crowding distance d_j means that solution lies in a less crowded area.



46



Create new children with "crowded tournament selection"

Crowded tournament selection

A solution i wins the tournament with another solution j if any of the following conditions are true:

1.If solution i lies on a "better" non-dominated front

2. If solutions i and j are on the same non-dominated rank but $d_i > d_j$

47

Summary

- Aggregation of several objectives into a single one requires setting of parameters.
- Why is Evolutionary computation good for multiobjective optimisation?
  - The solution space can be explored in a single optimization run
  - Applicable to complex and huge search spaces
  - Enables flexibility**:** problem formulation can be easily modified / extended.
- Elitist Non-dominating Sorting GA uses:
  - elitist principle
  - explicit diversity preserving mechanism
  - emphasises the non-dominated solutions

48