Chapter 1

Evolutionary Multiobjective Optimisation

1.1 Motivations

- Genetic Drift – Wikipedia
- Speciation – Wikipedia
- Punctuated Equilbria – Wikipedia

1.2 Implications for Evolutionary Optimisation

Implicit approaches

Impose an equivalent of geographical separation. Impose an equivalent of speciation

Explicit approaches

Make similar individuals compete for resources (fitness). Make similar individuals compete with each other for survival.

1.2.1 Explicit 1: Fitness Sharing

Restricts the number of individuals within a given niche by "sharing" their fitness, so as to allocate individuals to niches in proportion to the niche fitness. Need to set the size of the niche σ_{share}. In either genotype or phenotype space run EA as normal but after each gen set the shared fitness of each candidates with the equation of

$$f'(i) = \frac{f(i)}{\sum_{j=1}^{\mu} sh\big(d(i,j)\big)} \qquad sh(d) = \begin{cases} 1 - \dfrac{d}{\sigma}, & d < \sigma, \\ 0, & \text{otherwise.} \end{cases}$$

1.2.2 Explicit 2: Crowding

Attempts to distribute individuals **evenly** amongst niches. Relies on the assumption that offspring will tend to be close to parents. Uses a distance metric in phenotype/genotype space. Randomly shuffle and pair parents, then produce two offspring. Two parent/offspring tournaments - pair so that

$$d(p_1, o_1) + d(p_2, o_2) < d(p_1, o_2) + d(p_2, o_1)$$

1.3 Prerequisites

1.3.1 Ordered Set

Partial Ordered Set

Formally, a partial order is a **homogeneous binary relation** that is **reflexive**, **antisymmetric**, and **transitive**. A **partially ordered set** (*poset* for short) is an *ordered pair* $P = (X, \leq)$ consisting of a set X (called the *ground set* of P) and a partial order $\leq$ on X. When the meaning is clear from context and there is no ambiguity about the partial order, the set X itself is sometimes called a poset.

A **reflexive, weak,** or **non-strict partial order**, commonly referred to simply as a **partial order**, is a **homogeneous relation** $\leq$ on a set P that is **reflexive**, **antisymmetric**, and **transitive**. That is, for all $a, b, c \in P$, it must satisfy:

1. **Reflexivity:** $a \leq a$, i.e. every element is related to itself.
2. **Antisymmetry:** if $a \leq b$ and $b \leq a$ then $a = b$, i.e. no two distinct elements precede each other.
3. **Transitivity:** if $a \leq b$ and $b \leq c$ then $a \leq c$.

A non-strict partial order is also known as an **antisymmetric preorder**.

Total Order

In mathematics, a total order or linear order is a partial order ***in which any two elements are comparable***. That is, a total order is a binary relation $\leq$ on some set X, which satisfies the following for all a, b and c in X:

1. $a \leq a$ (*reflexive*).
2. If $a \leq b$ and $b \leq c$ then $a \leq c$ (*transitive*).
3. If $a \leq b$ and $b \leq a$ then $a = b$ (*antisymmetric*).
4. $a \leq b$ or $b \leq a$ (*strongly connected*, formerly called *totality*).

1.3.2 Cartesian Product

In **mathematics**, specifically **set theory**, the **Cartesian product** of two sets A and B, denoted $A \times B$, is the set of all **ordered pairs** (a, b) where a is an element of A and b is an element of B. In terms of **set-builder notation**, that is

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}.$$

Orders on the Cartesian product of totally ordered sets

There are several ways to take two totally ordered sets and extend to an order on the **Cartesian product**, though the resulting order may only be **partial**. Here are three of these possible orders, listed such that each order is stronger than the next:

- **Lexicographical order:** $(a, b) \leq (c, d)$ if and only if $a < c$ or ($a = c$ and $b \leq d$). This is a total order.
- $(a, b) \leq (c, d)$ if and only if $a \leq c$ and $b \leq d$ (the **product order**). This is a partial order.
- $(a, b) \leq (c, d)$ if and only if ($a < c$ and $b < d$) or ($a = c$ and $b = d$) (the reflexive closure of the **direct product** of the corresponding strict total orders). This is also a partial order.

Relations as Subsets of Cartesian Products Mathematics Exchange

The set of ordered pairs drawn from the Cartesian product can be defined as

$$R = \{(x, y) \in A \times B |\ xRy\}$$

Thus we can say

$$R \subseteq A \times B$$

1.4 Multiobjective Optimisation

Many problems have more than one goal function thus there is no single optimal function value. There is a general assumption that MO is as hard as Single-objective optimisation. We have the decision space of S^n with $s \in S^n$ and with our objective function $\mathsf{F}(f_1(s), f_2(s), \ldots, f_m(s))$ will yield the objective space of $\mathbb{R}^m$. Therefore, for a general MOP we have that

$$\mathsf{F} : S^n \to \mathbb{R}^m, \quad \text{where } n < m \vee n > m \vee n = m$$

We say $x \in \mathbb{R}^m$ dominates $y \in \mathbb{R}^m$ if it is at least as good on all criteria and better on at least one.

Dominance in the Objective Space

Assuming maximisation

x weakly dominates y ($x \succeq y$) iff $x_i \geq y_i \; \forall \, i \in \{1, \ldots, m\}$
x dominates y ($x \succ y$) iff $x \succeq y$ and $x \neq y$

This can be translated to the search space as

$$s \succeq s' \iff \mathsf{F}(s) \succeq \mathsf{F}(s')$$
$$s \succ s' \iff \mathsf{F}(s) \succ \mathsf{F}(s')$$

Pareto Front – Wikipedia

The Pareto Front consists of all the non-dominated objective vectors. Let $\boldsymbol{\pi} \subseteq \mathbb{R}^m$ be the set of feasible objective vectors such that

$$\boldsymbol{\pi} = \{\pi \in \mathbb{R}^m : \pi = \mathsf{F}(s) \text{ where } s \in S^n\}$$

For any objective vectors $\pi', \pi'' \in \boldsymbol{\pi}$, let $\mathsf{P}(\boldsymbol{\pi})$ be the Pareto set thus

$$\mathsf{P}(\boldsymbol{\pi}) = \left\{\pi' : \{\pi'' \succ \pi'\} = \emptyset\right\}$$

Parento Optimal

A solution $x^* \in X$ is then called *Parento Optimal* iff there is no other solution in X that dominates x^* [1].

Pareto set approximations

Specific sets of solutions are the so-called *Pareto set approximations*, which are solution sets of pairwisely non-dominated solutions [1].

EC for Parento Front

Population-based nature of search means you can simultaneously search for set of points approximating Pareto front. Don't have to make guesses about which combinations of weights might be useful. Don't have to make guesses about which combinations of weights might be useful.

1.4.1 NSGA-II

The non-dominated sorting GA-II is an elitist approach which rank and preserve the best solution on top. In order to sort a population of N size each solution must be compared with every other solution in the population to find if it is dominated. This requires $\mathcal{O}(MN)$ where M is the number of objective vectors. When this process is continued to find the members of the first non-dominated class for all population members the total complexity is $\mathcal{O}(MN^2)$. At this stage, all individuals in the first non-dominated front are found. In order to find the next front repeat the above procedure. As can be seen the worst case (when there exists only one solution in each front) complexity of this algorithm is $\mathcal{O}(MN^3)$. The fast NSGA-II will require at most $\mathcal{O}(MN^2)$ computations. In the following algorithm, it's worth mentioning that the space complexity is $\mathcal{O}(N^2)$. [2]. Please refer to the pseudo-code in the *lecture slide*.

1.4.2 Crowding Distance

To get an estimate of the density of solutions surrounding a particular point in the population we take the average distance of the two points on either side of this point along each of the objective vectors. This quantity i_{distance} serves as an estimate of the size of the largest cuboid enclosing the point i without including any other point in the population. The time complexity for the sorting is $\mathcal{O}(MN \log N)$ [2]. Please refer to the *lecture slide* for the pseudo-code and illustration.

1.4.3 Crowding Comparison Operator

The crowded comparison operator ($\succ_n$) guides the selection process at the various stages of the algorithm towards a uniformly spread out Pareto-optimal front. This helps to have a more diverse solution set [2].

1. Non-domination rank – i_{rank}
2. Local crowding distance – $i_{distance}$

$$i \succ_n j \text{ if } (i_{rank} < j_{rank}) \text{ or } \big((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance})\big)$$

1.4.4 Iteration for NSGA-II

Please refer to the *lecture slide* for the pseduo-code.

Initially, a random parent population ?p is created. The population is sorted based on the non-domination. Each solution is assigned a fitness equal to its non-domination level (1 is the best level). Thus, minimization of fitness is assumed. Binary tournament selection, recombination, and mutation operators are used to create a child population Q_0 of size N. Let us now look at the complexity of one iteration of the entire algorithm. The basic operations being performed and the worst-case complexities associated with them are as follows:

1. Non-dominated sort is $O(MN^2)$,
2. Crowding distance assignment is $O(MN \log N)$, and

3. Sort on $\geq_n$ is $O(2N \log(2N))$.

As can be seen, the overall complexity of the above algorithm is $\mathcal{O}(MN^2)$ [2].

1.4.5 Deteriorative Cycles

In single-objective optimization, every solution is mapped to a real value and solutions can always be pairwisely compared via the less or equal relation $\leq$ on $\mathbb{R}$. In another words, the total order (any elements are comparable) $\leq \subseteq \mathbb{R} \times \mathbb{R}$ induces via f order on the search space X that is a total preorder. In a multiobjective scenario, the $\leq$ relation is generalized to objective vectors, i.e., $\leq$ is a subset of $\mathbb{R}^k \times \mathbb{R}^k$. Here, the totality is not given due to vectors $a, b \in \mathbb{R}^k$ where $f_1(a) < f_1(b)$ but $f_2(a) > f_2(b)$—the relation $\leq$ on the set of objective vectors is only a partial order, i.e., reflexive, antisymmetric, and transitive. This means that *not any elements are comparable*. [1]

Downfall of The Evolutionary Algorithm Multiobjective Optimisation

Among the well-established ones, NSGA-II [Deb et al., 2002] and SPEA2 [Zitzler et al., 2002] have to be mentioned here. Both use the Pareto dominance concept as the main selection criterion in an elitist manner where non-dominated solutions are favored over dominated ones. In addition, a second selection criterion establishes diversity among the solutions. However, experimental studies have shown that both algorithms do not scale well if the number of objectives increases and that a cyclic behavior can be observed. This means that—although non-dominated solutions are preferred over dominated ones—over time, previously dominated solutions enter the population again, resulting in an oscillating distance to the Pareto front. [1]

1.4.6 Hypervolume Indicator

One of the latest areas of evolutionary multiobjective optimization where theoretical investigations have been made is the area of hypervolume-based search. The hypervolume indicator, initially proposed for performance assessment, has nowadays been used to guide the search in several multiobjective evolutionary algorithms such as the SMS-EMOA of Beume et al. [2007b], the multiobjective version of CMA-ES [Igel et al., 2007], or HypE [Bader and Zitzler, 2008, 2009]. As mentioned above, the hypervolume indicator is a unary quality measure mapping a set of solutions to the hypervolume in objective space that is dominated by the corresponding objective vectors of the solutions but not by a predefined set of *reference points*. More formally, for a given set $A \subseteq S^n$ of a solution set and set of reference points in the objective space $R \subseteq \mathbb{R}^m$. We define the set of objective vectors that are dominated by A but not by R as [1]

$$H(A, R) = \{h : \exists\, a \in A,\ \exists\, r \in R : \mathsf{F}(a) \leq h \leq r\}$$

And the corresponding hypervolume indicator with reference R is I_H^R as

$$I_H^R \coloneqq \mathrm{VOL}\left(\bigcup_{s \in A} \left[f_1(s), r_1\right] \times \left[f_2(s), r_2\right] \times \ldots \times \left[f_m(s), r_m\right]\right), \quad \text{where } \begin{cases} r_i \in R \subseteq \mathbb{R}^m \\ s \in A \end{cases}$$

This means the objective value with the corresponding reference point.

Chapter 2

Submodular function

Algorithm 1: Generalized Cost-benefit Algorithm

Data: $B > 0$
Result: Selection $S \subseteq V$
$v^* := \arg\max\{f(v) \mid v \in V, \hat{c}(v) \leq B\}$;
$G := \emptyset$;
$V' := V$;
while $V' \neq \emptyset$ **do**
  foreach $X \in V$ **do**
    $\Delta_f^X := f(G \cup X) - f(G)$;
    $\Delta_c^X := \hat{c}(G \cup X) - \hat{c}(G)$;
  $X^* = \arg\max \left\{ \Delta_f^X / \Delta_c^X \mid X \in V' \right\}$;
  if $\hat{c}(G \cup X^*) \leq B$ **then**
    $G := G \cup X^*$;
  $V' := V' \setminus X^*$;
return $\arg\max_{S \in \{v^*, G\}} f(S)$;

Algorithm 2: Generalized Greedy Algorithm

Input: Initial budget constraint B
Output: Selection $S \subseteq V$
$X \leftarrow \emptyset$;
$V' \leftarrow V$;
repeat
  $v^* \leftarrow \arg\max_{v \in V'} \frac{f(X \cup v) - f(X)}{\hat{c}(X \cup v) - \hat{c}(X)}$;
  if $\hat{c}(X \cup v^*) \leq B$ **then**
    $X \leftarrow X \cup v^*$;
  $V' \leftarrow V' \setminus \{v^*\}$;
until $V' = \emptyset$;
$v^* \leftarrow \arg\max_{v \in V, \hat{c}(v) \leq B} f(v)$;
return $\arg\max_{S \in \{X, \{v^*\}\}} f(S)$;

Approximate Local Search Procedure B:

Input: Ground set X of elements and value oracle access to submodular function f.

1. Let $\{v\}$ be a singleton set with the maximum value $f(\{v\})$ and let $S = \{v\}$.
2. While there exists the following delete or exchange local operation that increases the value of $f(S)$ by a factor of at least $1 + \frac{\epsilon}{n^4}$, then apply the local operation and update S accordingly.
   - **Delete operation on S.** If $e \in S$ such that $f(S \setminus \{e\}) \geq (1 + \frac{\epsilon}{n^4}) f(S)$, then $S \leftarrow S \setminus \{e\}$.
   - **Exchange operation on S.** If $d \in X \setminus S$ and $e_i \in S \cup \{\phi\}$ (for $1 \leq i \leq k$) are such that $(S \setminus \{e_1, \ldots, e_k\}) \cup \{d\} \in \mathcal{I}_i$ for all $i \in [k]$ and $f((S \setminus \{e_1, \ldots, e_k\}) \cup \{d\}) > (1 + \frac{\epsilon}{n^4}) f(S)$, then $S \leftarrow (S \setminus \{e_1, \ldots, e_k\}) \cup \{d\}$.

2.1 Global-SEMO

The use of one-bit mutations is attractive for theoretical investigations as its analysis is often easier compared to other mutation operators. It is, however, only a *local search operator* flipping one bit at a time and therefore does not guarantee a finite running time for all pseudo-Boolean functions. On the contrary, the independent-bit mutation operator, which flips each bit independently with a certain probability, can be called a *global search*

operator since it can produce any point in the decision space with positive probability. [3].

Algorithm 3: GSEMO (concise)

Input: Objective $g : \{0,1\}^n \to \mathbb{R}^m$
Output: Pareto set P
$x \sim \mathrm{Unif}(\{0,1\}^n)$;
$P \leftarrow \{x\}$;
repeat
  pick $x \in P$ u.a.r.; `// uniformly at random`
  $x' \leftarrow \mathrm{flip}_{1/n}(x_i)\ \forall i$; `// flip each bit independently w.p.` $1/n$
  compute $g(x')$;
  if $\nexists y \in P:\ g(y) \prec g(x')$ **then**
    $P \leftarrow (P \cup \{x'\}) \setminus \{y \in P:\ g(y) \preceq g(x')\}$;
until *stop*;
return P

2.1.1 Knapsack Problem on GSEMO [3]

The optimization problem considered in this study is a special instance of the multiobjective knapsack problem. Given is a set of n items, each of which has m profit and k weight values associated with it. The goal is to select a subset of items such that the sums over each of their k-th weight values do not exceed given bounds and the sums over each of their m-th profit values are maximized.

Definition 1 (MOKP)**.** *Let $X = \{0,1\}^n$ be the decision space and $n \in \mathbb{N}$. The Multiobjective Knapsack Problem (MOKP) is*

$$\textit{maximize} \quad f(x) = \big(f_1(x), f_2(x)\big) \qquad \textit{subject to} \quad g(x) \leq \tfrac{n}{2},$$

with

$$f_1(x) = \sum_{i=1}^{n} 2^{n-i} x_i, \qquad f_2(x) = \sum_{i=1}^{n} 2^{i-1} x_i, \qquad g(x) = \sum_{i=1}^{n} x_i, \qquad x = (x_1, \ldots, x_n) \in X^n.$$

Pareto set size and structure. The MOKP function is a bijection, so none of the $|X| = \Theta(2^n)$ decision vectors maps to the same objective vector. The Pareto set contains $|X^*| = n^2/4 + 1$ elements and has the form below.

Proposition 1. *A decision vector $x \in X$ is Pareto-optimal in the MOKP problem if and only if it has the form*

$$\underbrace{(1,1,\ldots,1}_{a},\underbrace{0,0,\ldots,0}_{n/2},\underbrace{1,1,\ldots,1)}_{n/2-a} \qquad \textit{(Type 1)}$$

or

$$\Big(\underbrace{1,1,\ldots,1}_{b},\underbrace{0,0,\ldots,0}_{c},1,\underbrace{0,0,\ldots,0}_{\frac{n}{2}-c},\underbrace{1,1,\ldots,1}_{\frac{n}{2}-b-1}\Big), \qquad \textit{(Type 2)}$$

where $a \in \{0,1,\ldots,n/2\}$, $b \in \{0,1,\ldots,n/2-1\}$, and $c \in \{1,\ldots,n/2-1\}$.

Informal proof sketch. Firstly only the decision vector with size $\leq 1/2$ can be inside the Pareto-optimal set. Thus there exists a vector with $p, q \in \{2,3,\ldots,n\}$ where $p > q$ such that $x_p = x_q = 1$ and $x_{p-1} = x_{q+1} = 0$. By inverting those bits, for the first objective function we have the gain $2^{n-p} - 2^{n-q-1} > 0$, and for the second objective function we have the gain $2^{q-1} - 2^{p-2} > 0$. As the first objective favors small indices and the second favors large indices, maximizing one requires compensating in the other; hence the one-bit is pushed toward both sides. □

Supported solutions. Only the $n/2+1$ Pareto-optimal decision vectors of the first type are 'supported' solutions. A solution is supported if there exists a weight combination such that it maximizes the weighted sum $\sum_{i=1}^{m} w_i f_i(x)$ (a single-objective surrogate).

Proposition 2. *Let* $\Phi(x) \stackrel{def}{=} w f_1(x) + (1-w) f_2(x)$*, where* $w \in [0,1]$*. For each Pareto-optimal decision vector* x'' *of Type 2 and each* $w \in [0,1]$*, there exists some Pareto-optimal vector* x' *of Type 1 such that* $\Phi(x') > \Phi(x'')$*.*

Proof. Let w.l.o.g. x'' be a Pareto-optimal vector of Type 2, where the bit with index $b+c+1$ is set to one. Let x', x''' be defined as the two vectors identical with x'' at $n-2$ positions, where only this one-bit is moved to position $b+1$ and $n/2+b+1$, respectively. Clearly, these x', x''' belong to Type 1. We show that $(\Phi(x')+\Phi(x'''))/2 > \Phi(x'')$, which implies that at least one of $\Phi(x')$ and $\Phi(x''')$ is larger than $\Phi(x'')$:

$$
\begin{aligned}
& (\Phi(x') + \Phi(x'''))/2 > \Phi(x'') \\
\Leftrightarrow\ & w\Big(2^{\,n-b-2} + 2^{\,n-n/2-b-2}\Big) + (1-w)\Big(2^{\,b-1} + 2^{\,n/2+b-1}\Big) > w\,2^{\,n-b-c-1} + (1-w)\,2^{\,b+c} \\
\Leftrightarrow\ & w\Big(2^{-1} + 2^{-n/2-1}\Big) + (1-w)\Big(2^{-1} + 2^{\,n/2-1}\Big) > w\,2^{-c} + (1-w)\,2^{\,c}.
\end{aligned}
$$

The last inequality holds for $1 \le c \le n/2-1$, which covers all possible Type 2 vectors by Proposition 1. □

Remark (On the choice of x' and x'''). I am considering why the author uses the first 0 bit in the middle section for Type 1 (in x') and the last 0 bit before the right section of 1s (in x'''). I am guessing that this is because these positions produce the smallest possible maximization for one objective while minimizing the gain in the other. On the left block of 1s, the outermost position yields the least value for f_1 (due to exponential decay), so moving the isolated 1 to the start of the middle 0s (for x') provides only a modest boost to f_1 but significantly reduces f_2. Similarly, for x''', moving it to the end of the middle 0s (just before the right 1s) modestly boosts f_2 but greatly penalizes f_1. The logic is symmetric for the second objective. These are the "minimal improvement" moves closest to x''; if $(\Phi(x') + \Phi(x'''))/2 > \Phi(x'')$ already holds here, it holds for any farther move (which improves even more), confirming that no Type 2 vector can be supported.

References

[1] Dimo Brockhoff. *Theoretical Aspects of Evolutionary Multiobjective Optimization—A Review*. Research Report 7030. Inria Research Report RR-7030. France: Inria, Sept. 2009.

[2] Kalyanmoy Deb et al. *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II*. Tech. rep. KanGAL Report No. 200001. Kanpur, India: Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology Kanpur, 2000. URL: http://www.iitk.ac.in/kangal.

[3] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. "Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem". In: *Natural Computing* 3.1 (2004), pp. 37–51. DOI: 10.1023/B:NACO.0000023415.41374.54.