UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Laurea triennale in Informatica
Anno accademico 2021/2022

Fondamenti di Intelligenza Artificiale

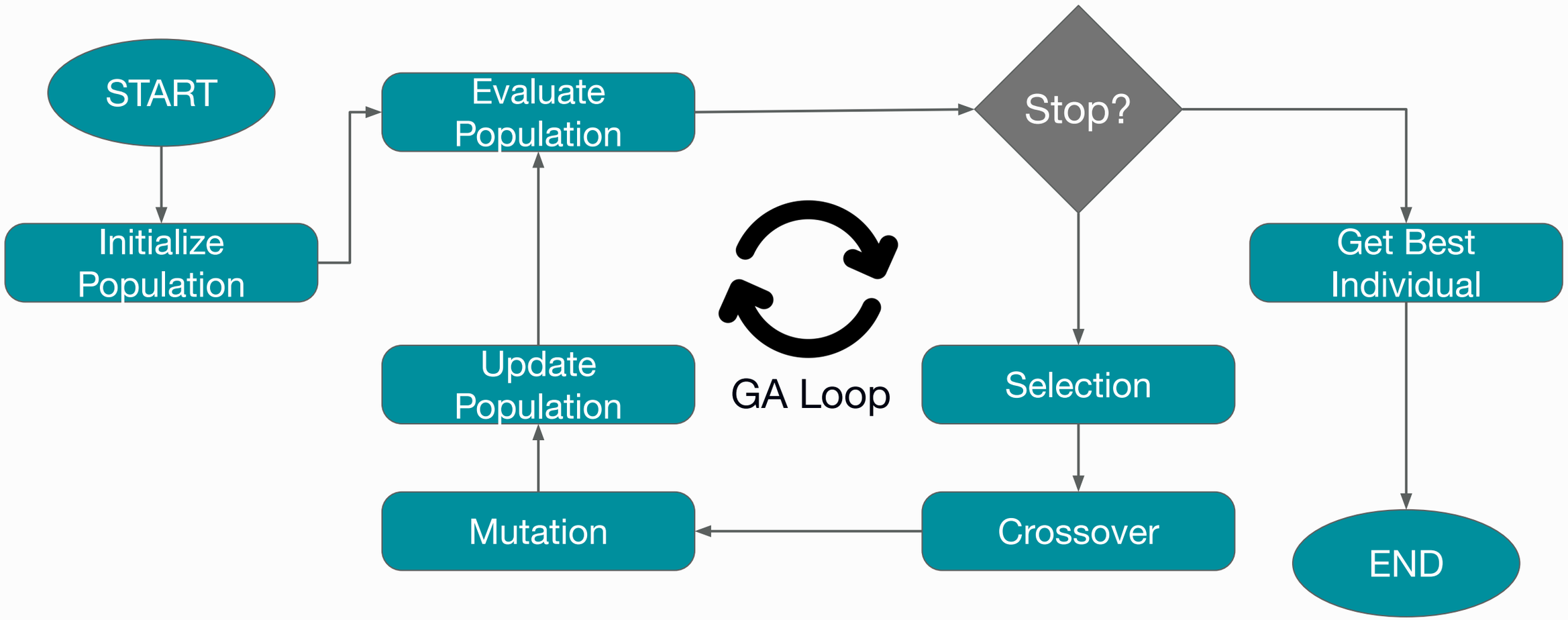
Lezione 8 - Algoritmi di ricerca locale (III)

Algoritmi Genetici

La scorsa lezione...

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

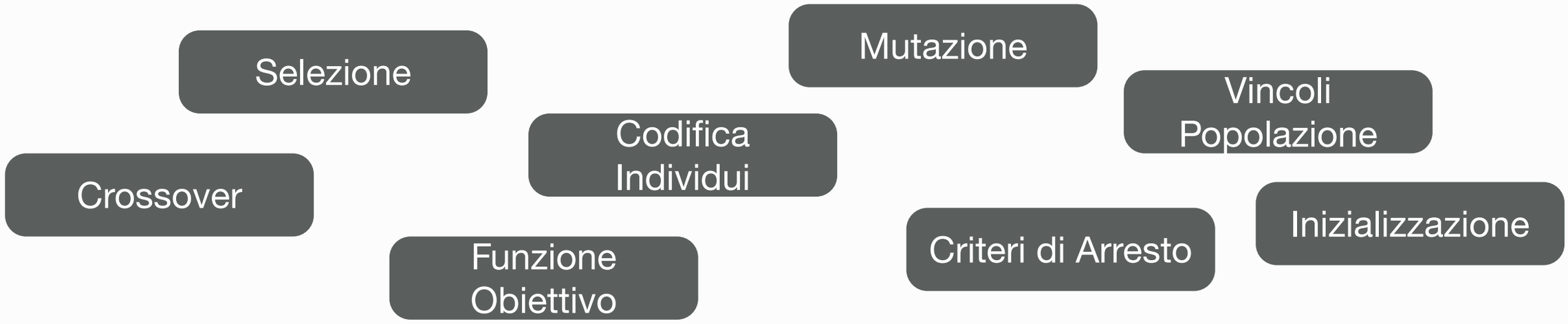
*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*



Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:



Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

L’operatore di selezione è ciò che realizza il meccanismo della **selezione naturale** nella sua essenza: favorire gli individui più promettenti (quelli con alto punteggio di valutazione) e sfavorire degli altri (quelli con basso punteggio).

Come già visto, la selezione non avviene usando *direttamente* il punteggio di valutazione, ma attraverso un **punteggio di fitness**, calcolato usando una funzione di **funzione di fitness.** La Roulette Wheel Selection adotta una funzione di fitness basata sulla valutazione *relativa* di ciascun individuo rispetto agli altri (*fitness-proportionate*).

Una selezione di tipo fitness-proportionate porta con sé alcuni rischi. Infatti, con popolazioni piccole si rischia di favorire troppo i migliori individui, che verrebbero selezionati molte volte, andando via via a **perdere diversità nella popolazione.**

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

E’ necessariamente un male? Gli algoritmi genetici fanno della *diversità* un loro punto di forza: senza di essa non avremmo sufficiente “forza di ricerca globale”, rischiando di degenerare in una ricerca locale classica (tanto vale usare Hill Climbing).

La perdita di diversità, infatti, ha una conseguenza negativa. Quando gli individui in una popolazione iniziano a somigliarsi troppo, non è difficile che il crossover dia luogo a *figli geneticamente simili ai genitori*, riducendo le probabilità di essere migliori di loro. Questo fenomeno prende il nome di **convergenza prematura**, che si può osservare quando l’evoluzione smette di dare miglioramenti (o comunque trascurabili).

Esistono diverse strategie adottate *su più livelli* atte a ridurre questo rischio. Le principali si adottano con un buon algoritmo di selezione, in grado di fornire una giusta **pressione di selezione**: favorire gli individui forti ma senza trascurare quelli meno forti.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Selezione

Mutazione

Vincoli Popolazione

Codifica Individui

Crossover

Inizializzazione

Criteri di Arresto

Funzione Obiettivo

Rank Selection

Si valutano tutti gli individui e si ordinano rispetto al punteggio di valutazione ricevuto. A ciascuno viene assegnata la posizione nell'ordinamento: il *rango*. Dopodiché, la funzione di fitness assegna a ciascun individuo una probabilità di selezione **proporzionata al rango**, così calcolata: $fit(x) = (|P| - rank(x) + 1) / \Sigma\, rank(x)$

E' applicabile anche con **valutazioni negative** (impossibile con Roulette Wheel per via della somma presente al denominatore). Mitiga parzialmente il problema delle convergenza prematura, andando ad "appiattire le differenze" tra le valutazioni dei diversi individui. In termini computazionali costa di più della Roulette Wheel per via dei continui ordinamenti da applicare alle varie generazioni.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

K-way Tournament Selection

Fissato un valore $1 < K < |P|$ arbitrario, si selezionano casualmente K individui per formare un cosiddetto “torneo”, all’interno del quale il migliore (con valutazione massima) passerà la selezione. Si ripete per $M < |P|$ (valore arbitrario) tornei, quindi fino ad ottenere un totale di M individui selezionati.

Non richiede alcun ordinamento ed è applicabile anche con valutazioni negative. Incarna in maniera fedele il concetto della “legge del più forte”. E’ una scelta molto popolare, ma impone l’onore di dover **scegliere i parametri K ed M**. Esistono delle varianti, ad esempio, invece di far vincere sempre il migliore si applica una *mini-Roulette Wheel*, oppure si impediscono *vittorie multiple* di uno stesso individuo.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Al di là dell'algoritmo di selezione, c'è sempre il rischio che la selezione vada a **sfavorire individui molto buoni**. Per quanto sia la natura a funzionare così, ai nostri fini non è una scelta furba perdere "a cuor leggero" soluzioni promettenti.

Elitism

Una sorta di "fattore correttivo" che impone il salvataggio di un sottoinsieme di individui composto dai migliori (secondo i punteggi di valutazione) direttamente nella generazione successiva, **bypassando il processo di selezione**. Questi individui prendono il nome di *elite*, che di solito non supera il 10% dell'intera popolazione.

Canonicamente, l'elite non deve partecipare né al crossover e né alla mutazione, ma niente ci vieta di farlo ugualmente e verificare empiricamente i risultati.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

A seconda dell'algoritmo di selezione, il mating pool può avere una dimensione diversa. Con algoritmi di **selezione con ricampionamento** (quelli in cui un individuo può passare più volte la selezione) il mating pool può essere grande quanto la popolazione.

Con algoritmi di **selezione senza ricampionamento** (che non abbiamo visto) il mating pool deve essere necessariamente di dimensione inferiore. Chiaramente, in tal caso avremmo una nuova generazione più piccola di quella precedente, andandosi via via a ridurre a mano a mano che l'algoritmo avanza.

Una soluzione sarebbe quella di effettuare una prima selezionare di $|M| < |P|$ individui, generare gli $|M|$ figli, e rieffettuare una seconda selezione (anche con un diverso algoritmo) tra tutti i $2|M|$ individui. In questo scenario **i figli non rimpiazzano automaticamente i genitori**.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

L’operatore di crossover rappresenta il primo passo per la creazione di nuove soluzioni per **esplorare** aree sconosciute dello spazio di ricerca.

Il crossover è l’operatore più sensibile alla codifica scelta per gli individui. Infatti, la decisione della codifica deve essere sempre valutata sulla base dell’esistenza di un operatore di **crossover sensato**. Qualora non dovesse esistere già un algoritmo di crossover noto, è comunque possibile *definirlo ad-hoc* per lo specifico problema.

Vediamo i principali operatori per individui codificati come array (di bit o di numeri).

Algoritmi Genetici

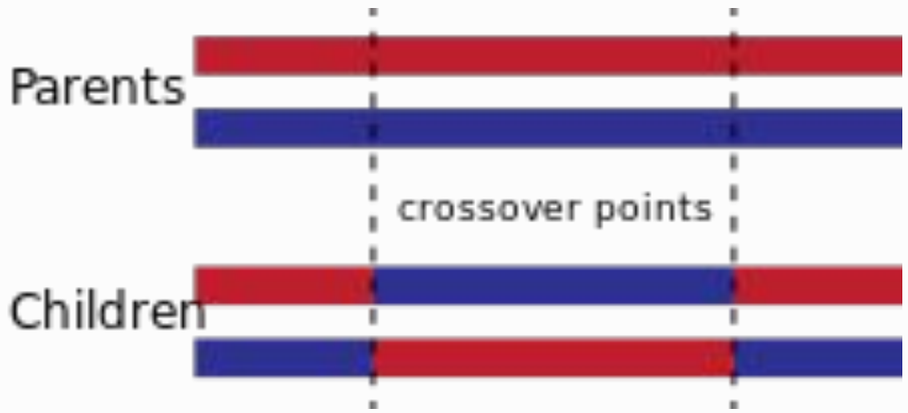
Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Selezione
Mutazione
Vincoli Popolazione
Codifica Individui
Crossover
Inizializzazione
Funzione Obiettivo
Criteri di Arresto

Two-Point Crossover

Invece di scegliere un singolo punto di taglio se ne scelgono due. Crea sicuramente una maggiore diversità, ed è possibile generalizzarlo a K punti. Un numero troppo alto non va bene perché potrebbe portare **troppa** diversificazione.



Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Selezione
Mutazione
Vincoli Popolazione
Codifica Individui
Crossover
Inizializzazione
Funzione Obiettivo
Criteri di Arresto

Uniform Crossover

Non si compie alcun taglio, ma il gene i-esimo di entrambi i figli sarà pari al gene i-esimo di uno dei due genitori. Dato che la scelta tra i due genitori è uniforme, si avranno figli con circa 50% del materiale del primo genitore e 50% del secondo.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Come visto nell'esempio della scatola nera, il crossover non si applica ciecamente ad ogni coppia di genitori, ma bisogna sempre verificare una **probabilità di crossover** (p_c), decisa a priori. Questa scelta viene fatta per evitare di creare una popolazione di figli troppo diversa dai genitori, aumentando troppo l'exploration a scapito dell'exploitation.

Di solito questa probabilità è abbastanza alta, ad esempio, 0.8. Inoltre, se esiste un meccanismo di identificazione della convergenza prematura, è possibile **modificarla dinamicamente** (*Adaptive GA*), magari aumentandola, per ridurne l'effetto.

Nella maggior parte dei casi il crossover avviene usando due genitori, ma esistono studi che hanno considerato il **multi-parent crossover**, che utilizza più di due individui per la generazione dei figli. Non è una scelta molto popolare, però potrebbe risultare utile in alcune classi di problemi.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

L’operatore di mutazione è il secondo passo per la creazione di nuove soluzioni per **esplorare** aree sconosciute dello spazio di ricerca.

Sebbene in misura minore, la mutazione è sensibile alla codifica scelta per gli individui. Infatti, la decisione della codifica deve essere valutata anche in base dell’esistenza di un operatore di mutazione sensato. Qualora non dovesse esistere già una mutazione nota, è comunque possibile *definirla ad-hoc* per lo specifico problema.

La mutazione è considerato un “meccanismo di sicurezza” che permette al GA di sbloccarsi da situazioni di convergenza prematura. Il parametro di governo p_m indica la **probabilità di mutazione** di un gene ed è tendenzialmente basso (0.01) proprio per evitare di degenerare in una ricerca casuale. Come per il crossover, è possibile modificare questa probabilità dinamicamente in caso di convergenza prematura.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Random Resetting Modificare casualmente un gene ad un altro valore.

Swap Scambiare di posto due geni scelti casualmente.

Scramble Selezionare una sottosequenza casuale di geni e la si permuta casualmente.

Inversion Selezionare una sottosequenza casuale di geni e la si inverte.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Gli operatori di crossover e mutazione creano nuove soluzioni. Finora abbiamo sempre considerato “lecite” le soluzioni create, ma in realtà i problemi di ottimizzazioni possono definire un **insieme di vincoli** (*constraints*), ovvero delle condizioni che le soluzioni devono necessariamente rispettare. I vincoli, infatti, definiscono quel sottoinsieme del dominio che prende il nome di **regione ammissibile** (*feasible region*).

Come si fanno a rispettare i vincoli nonostante l’imprevedibilità del crossover e della mutazione? Esistono **diversi metodi**:

Preservare Ammissibilità

Partendo da soluzioni ammissibili (*feasible*), si mettono in campo degli accorgimento affinché gli operatori di crossover e mutazione non creino mai soluzioni inammissibili (*infeasible*), quindi facendo in modo di **mantenere sempre validi tutti i vincoli definiti**.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Gli operatori di crossover e mutazione creano nuove soluzioni. Finora abbiamo sempre considerato “lecite” le soluzioni create, ma in realtà i problemi di ottimizzazioni possono definire un **insieme di vincoli** (*constraints*), ovvero delle condizioni che le soluzioni devono necessariamente rispettare. I vincoli, infatti, definiscono quel sottoinsieme del dominio che prende il nome di **regione ammissibile** (*feasible region*).

Come si fanno a rispettare i vincoli nonostante l’imprevedibilità del crossover e della mutazione? Esistono **diversi metodi**:

Funzione di Penalità

Modificare la funzione di valutazione aggiungendo una componente di **penalità**, che riduce il punteggio in base a “quanto si viola” un vincolo. Bisogna stabilire una misura che catturi il **grado di violazione**.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Gli operatori di crossover e mutazione creano nuove soluzioni. Finora abbiamo sempre considerato “lecite” le soluzioni create, ma in realtà i problemi di ottimizzazioni possono definire un **insieme di vincoli** (*constraints*), ovvero delle condizioni che le soluzioni devono necessariamente rispettare. I vincoli, infatti, definiscono quel sottoinsieme del dominio che prende il nome di **regione ammissibile** (*feasible region*).

Come si fanno a rispettare i vincoli nonostante l’imprevedibilità del crossover e della mutazione? Esistono **diversi metodi**:

Favorire Soluzioni Ammissibili

Ipotizzando una Tournament Selection, le soluzioni ammissibili vincono automaticamente contro le inammissibili. Tra due soluzioni inammissibili vince quella con un numero minore di violazioni. Chiaramente, tra due ammissibili si applicano le solite regole.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Abbiamo già visto alcuni dei principali criteri di arresto dell'evoluzione, così come abbiamo detto che è possibile creare delle condizioni composte (legate da OR).

Basate su Budget di Ricerca

L'evoluzione termina se l'algoritmo spende tutte le risorse ad esso allocate.

Tempo di Esecuzione

L'evoluzione termina se l'algoritmo è in esecuzione da più di X secondi. E' un criterio che conviene sempre considerare come ultima spiaggia.

Funzione di Costo

Si definisce una **funzione di costo**, che assegna a ciascun individuo un valore di costo, indicante il "consumo" di quell'individuo rispetto ad una certa risorsa. Se si ottiene una popolazione la cui somma dei costi supera il costo massimo allocato X, l'evoluzione termina.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Abbiamo già visto alcuni dei principali criteri di arresto dell'evoluzione, così come abbiamo detto che è possibile creare delle condizioni composte (legate da OR).

Basate su Budget di Ricerca L'evoluzione termina se l'algoritmo spende tutte le risorse ad esso allocate.

Numero Generazioni L'evoluzione termina se l'algoritmo ha creato più di X generazioni.

Numero Valutazioni L'evoluzione termina se l'algoritmo ha eseguito più di X volte la funzione di valutazione.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Abbiamo già visto alcuni dei principali criteri di arresto dell'evoluzione, così come abbiamo detto che è possibile creare delle condizioni composte (legate da OR).

Basate su Criteri di Convergenza — L'evoluzione termina se la popolazione ha raggiunto una situazione di convergenza da cui non è possibile uscirne.

Convergenza Fenotipica — L'evoluzione termina se dopo X generazioni non si osservano miglioramenti rilevanti. Per valutare l'intera popolazione si può usare la *valutazione media*: i miglioramenti sono "rilevanti" se la differenza supera una soglia arbitraria ε.

Convergenza Genotipica — L'evoluzione termina se gli individui si somigliano per un $X\%$ (ovvero, $X\%$ dei loro geni sono perfettamente identici). Di solito X viene fissato a 90.

Test Ottimalità — Se disponibile, si può usare un test di ottimalità per verificare il raggiungimento dell'ottimo, così da arrestarsi prima del tempo.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

Si è visto empiricamente come la popolazione iniziale influenzi in maniera rilevante le prestazioni di ricerca di un GA. In particolare, ci si aspetta che quanto più la popolazione sia diversificata tanto più è facile trovare buone soluzioni.

Spesso si crea la prima popolazione in maniera del tutto **casuale** (preferibilmente evitando individui inammissibili in problemi vincolati), ma esistono altre tecniche più sofisticate basate sull'uso di **metriche di diversità**, che fanno in modo che la popolazione iniziale inizia già con un set di individui ben diversi.

La metrica si può basare sulla **dispersione fenotipica** (ad esempio, deviazione standard delle valutazioni degli individui), altre sulla **dispersione genotipica** (ad esempio, il grado di somiglianza delle codifiche degli individui).

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:

In genere un GA utilizza una **popolazione a taglia fissa**, determinata da un parametro arbitrario deciso a priori. Alcune implementazioni prevedono una **popolazione di taglia variabile** tra le generazioni (di nuovo, *Adaptive GA*), ma sono più complesse da gestire.

Popolazioni troppo grandi danneggiano le prestazioni, mentre quelle piccole rischiano facilmente di convergere prematuramente.

Sempre per ragioni di performance si potrebbero imporre altri limiti alla popolazione oltre la taglia, ad esempio impostando un **limite di costo** per evitare di danneggiare le prestazioni (sulla falsariga del *problema dello zaino*).

Utilizzando **euristiche problem-specific** si potrebbero scartare individui poco promettenti e generarne di nuovi.

Algoritmi Genetici

Problemi Multi-Obiettivo

Algoritmi Genetici

Problemi Multi-Obiettivo

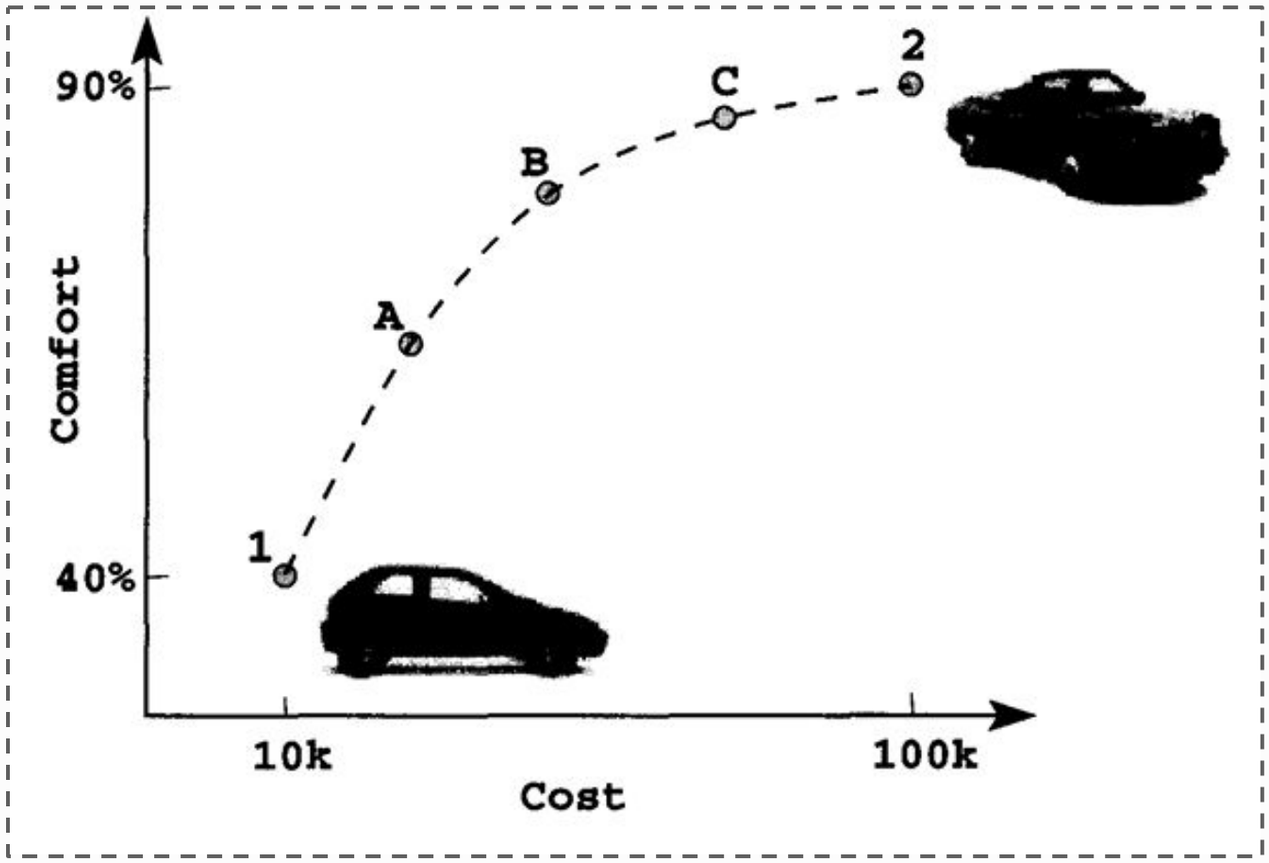
Finora abbiamo visto l'ottimizzazione di un singolo obiettivo. La realtà, però, è molto più complessa. Prende le decisioni, spesso, è una questione di **trade-off**: bisogna trovare un compromesso, un equilibro tra **più obiettivi contrastanti**.

Per fare un esempio semplice, immaginiamo di dover acquistare un'automobile. Sappiamo bene che esistono molti criteri di acquisto definiti sulla base delle nostre preferenze/esigenze. Supponiamo di dover acquistare la migliore automobile offerta da un rivenditore tale che sia (1) di prezzo vantaggioso, e (2) quanto più comoda possibile.

Se volessimo **massimizzare il comfort**, allora l'*auto #2* sarebbe la scelta ottimale, ma se volessimo **minimizzare il costo** la scelta migliore sarebbe chiaramente l'*auto #1*.

Se volessimo considerare in maniera equa entrambi i criteri, otteniamo **un insieme di soluzioni candidate ad essere le migliori.**

Sorge spontanea la domanda: ma tra queste cinque auto, qual è la migliore?



Algoritmi Genetici

Problemi Multi-Obiettivo

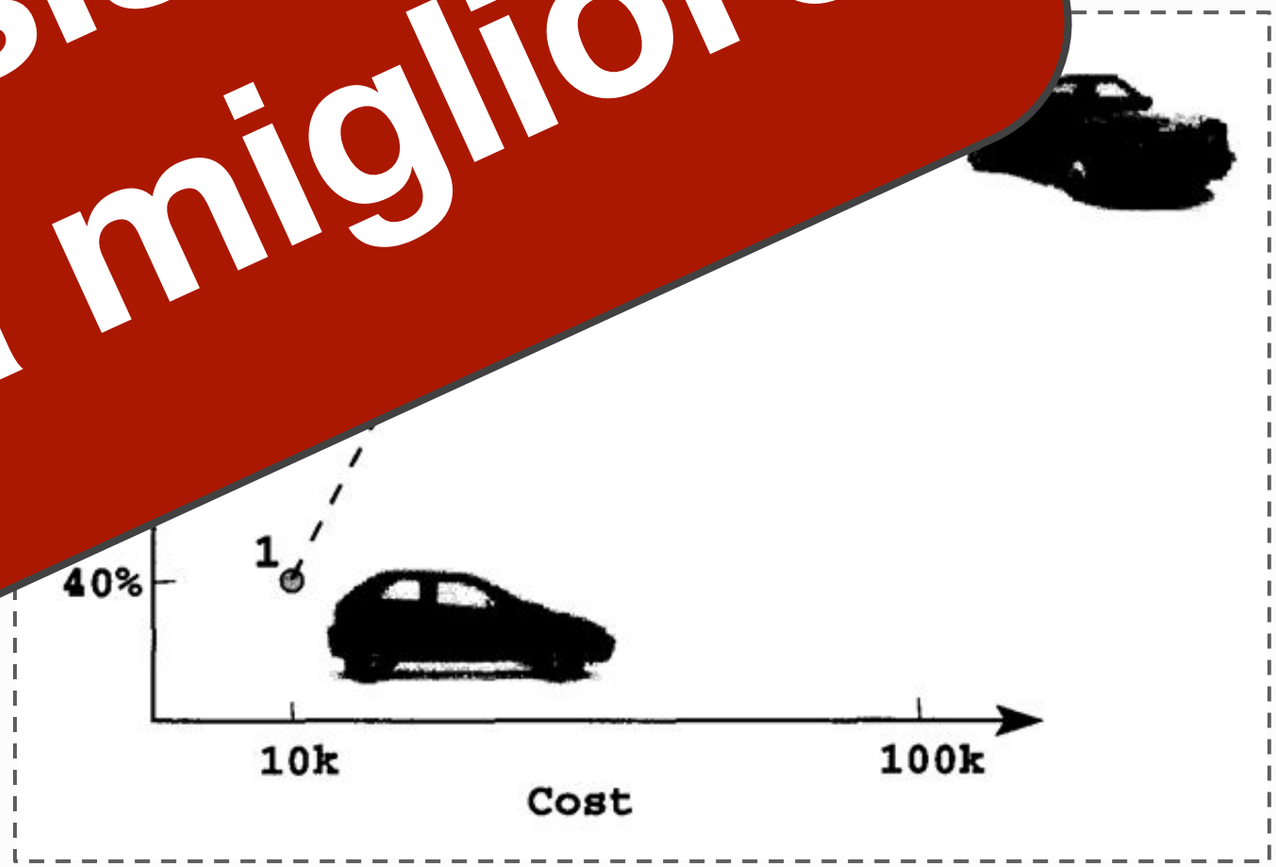
Finora abbiamo visto l'ottimizzazione di un singolo obiettiv erò, è molto più complessa. Prende le decisioni, spesso, è una questi ogna trovare un compromesso, un equilibro tra **più obiettivi**.

Per fare un esempio semplice, immagini bile. Sappiamo bene che esistono mol ostre preferenze/esigenze. Suppo fferta da un rivenditore tale ch ssibile.

Se volessim allora l'a ottimal **il costo** chiarame

Se volessi equa entram **insieme di so essere le migl**

Sorge spontanea da: ma tra queste cinque auto, qual è la migliore?



Non esiste LA scelta migliore

Algoritmi Genetici

Problemi Multi-Obiettivo

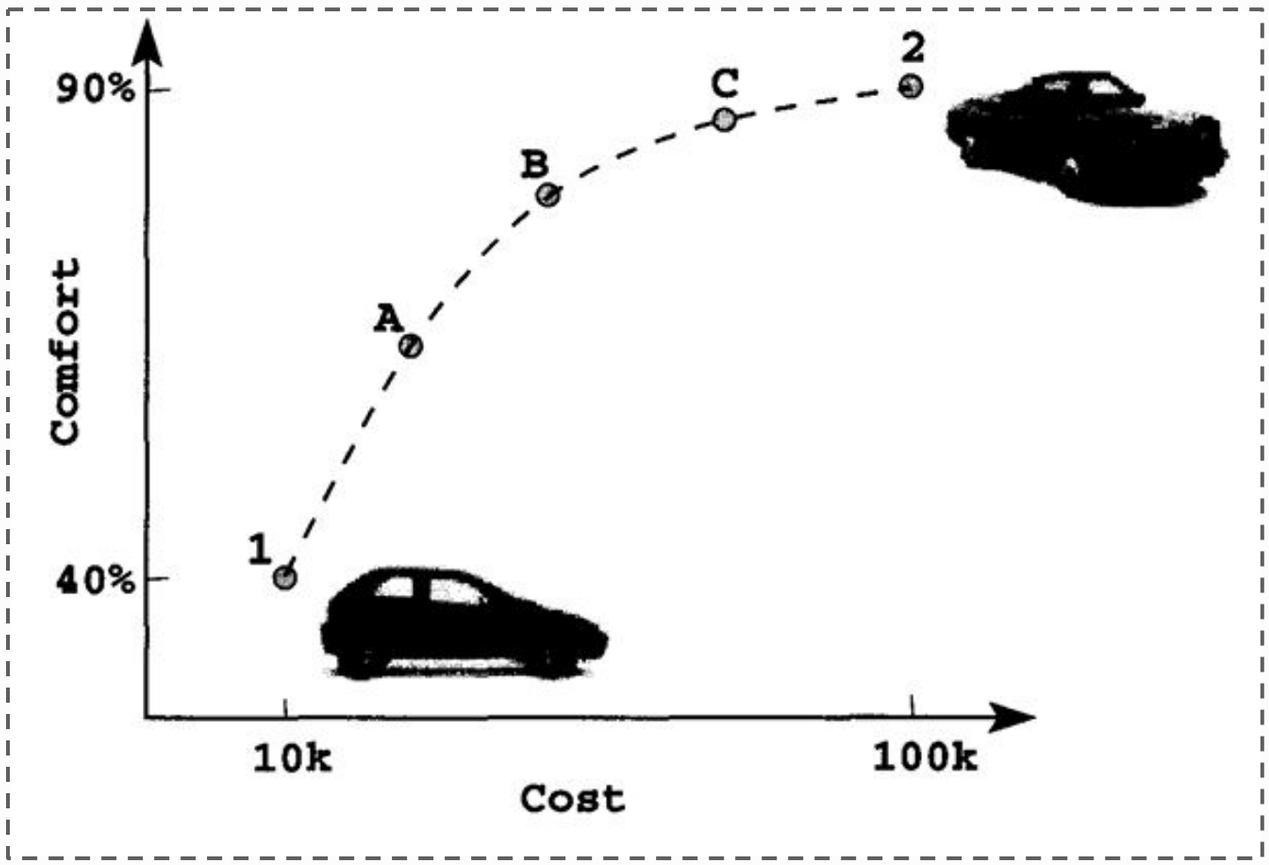
Finora abbiamo visto l'ottimizzazione di un singolo obiettivo. La realtà, però, è molto più complessa. Prende le decisioni, spesso, è una questione di **trade-off**: bisogna trovare un compromesso, un equilibro tra **più obiettivi contrastanti**.

Per fare un esempio semplice, immaginiamo di dover acquistare un'automobile. Sappiamo bene che esistono molti criteri di acquisto definiti sulla base delle nostre preferenze/esigenze. Supponiamo di dover acquistare la migliore automobile offerta da un rivenditore tale che sia (1) di prezzo vantaggioso, e (2) quanto più comoda possibile.

Se volessimo **massimizzare il comfort**, allora l'*auto #2* sarebbe la scelta ottimale, ma se volessimo **minimizzare il costo** la scelta migliore sarebbe chiaramente l'*auto #1*.

La differenza cruciale tra ottimizzazione mono-obiettivo ed ottimizzazione multi-obiettivo (MOO) è nel numero di soluzioni ottime: i MOO prevedono un **set di soluzioni ottimali non confrontabili tra loro**.

Sorge spontanea la domanda: ma tra queste cinque auto, qual è la migliore?



Algoritmi Genetici

Problemi Multi-Obiettivo

E’ lecito chiedersi: *“cosa me ne faccio di tutte queste soluzioni ottimali?”*

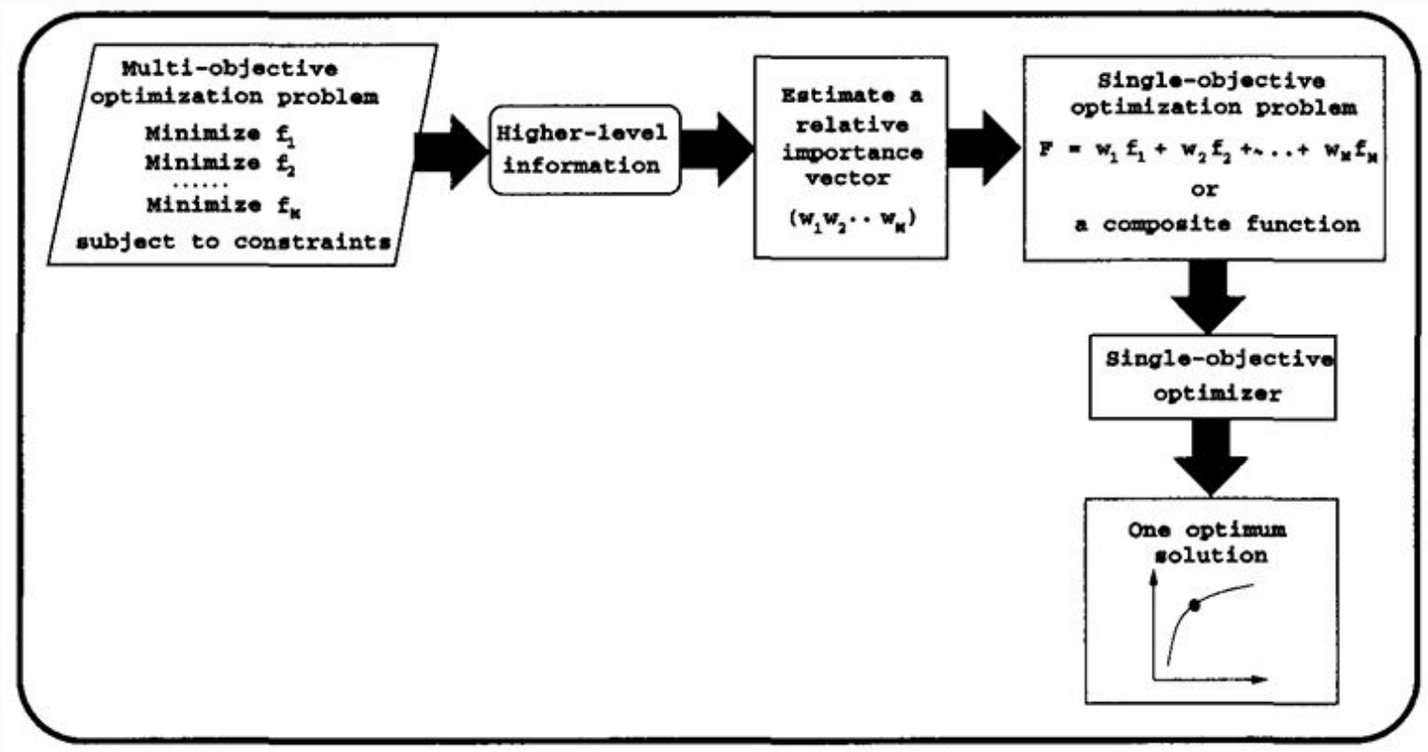
Esistono sostanzialmente **due approcci** per poter gestire i problemi multi-obiettivo:

Approccio Classico

Convertire il problema multi-obiettivo in un problema mono-obiettivo **aggregando tutte le funzioni obiettivo in una singola**, tipicamente tramite una **somma pesata.** Il vettore dei pesi viene stabilito sulla base di **informazioni di alto livello provenienti dallo specifico problema** che si vuole risolvere. Ad esempio, se il costo e il comfort sono ugualmente importanti, allora si usa un vettore $<0.5, 0.5>$.

Semplice da realizzare poiché riusa gli ottimizzatori classici, ma…

- la scelta dei pesi è molto soggettiva;
- si perde il concetto di *trade-off*, a cui magari siamo interessati;
- Le funzioni obiettivo possono essere su scala e ordini differenti, come nel caso di comfort e costo.



Algoritmi Genetici

Problemi Multi-Obiettivo

E’ lecito chiedersi: *“cosa me ne faccio di tutte queste soluzioni ottimali?”*

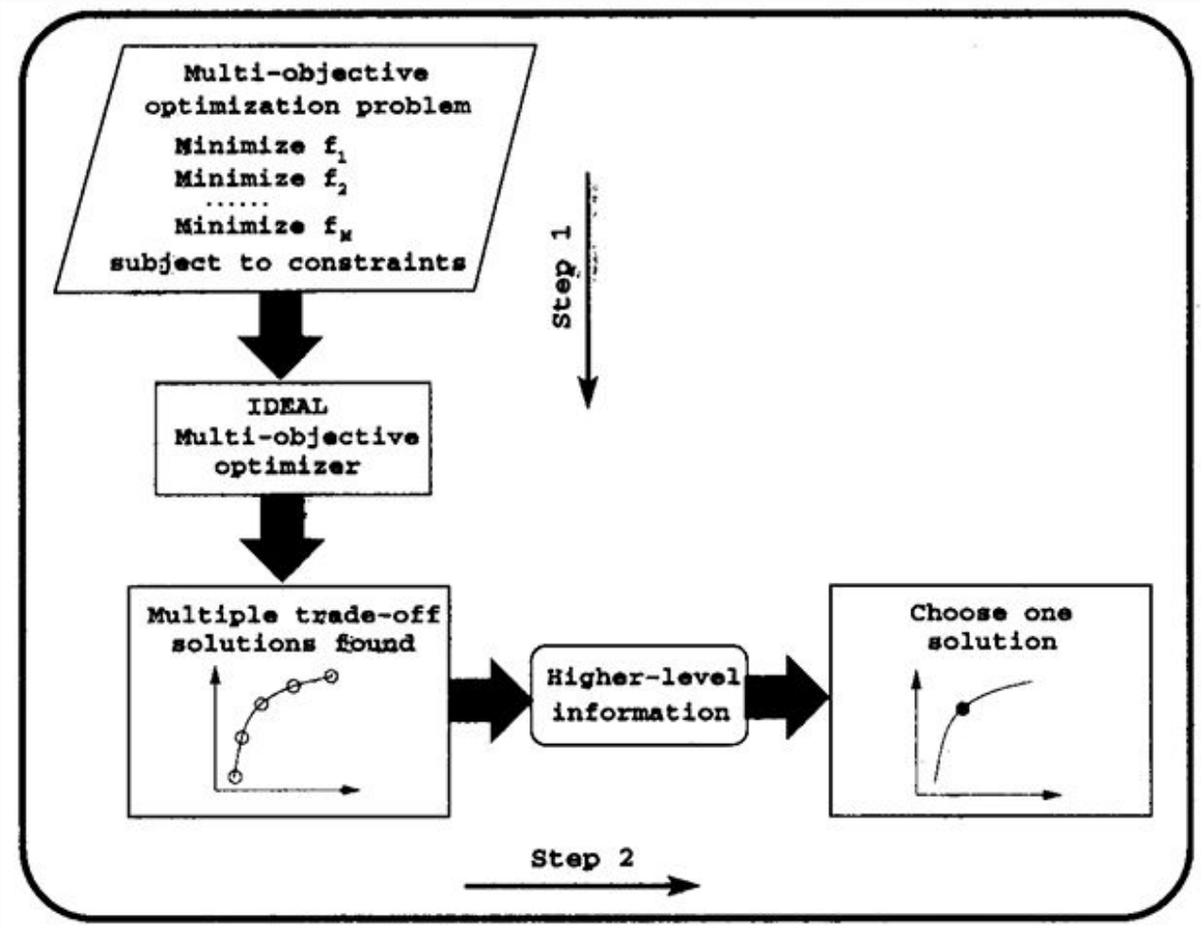
Esistono sostanzialmente **due approcci** per poter gestire i problemi multi-obiettivo:

Approccio Ideale

Ottimizzare il problema tenendo conto dei trade-off di tutte le funzioni obiettivo. L’ottimizzazione produce un **set di soluzioni ottimali**, dal quale, in un secondo momento, si **identifica la soluzione che fa al caso nostro** tramite informazioni di alto livello provenienti dallo specifico problema.

Più complesso da realizzare poiché richiede la definizione di nuovi tipi di ottimizzatori, ma supera i problemi evidenziati dall’approccio classico. Chiaramente, esploreremo questo approccio.

La vera difficoltà risiede principalmente nel trovare questo set di soluzioni ottimali.



Algoritmi Genetici

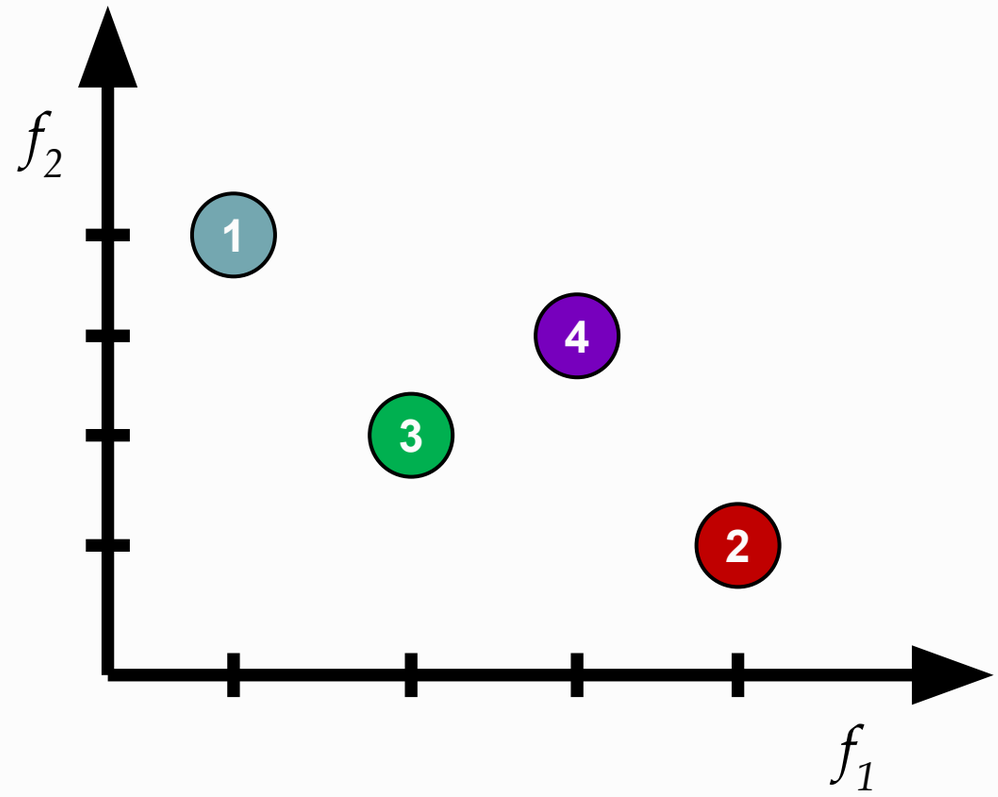
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

Ad ogni soluzione, quindi, non viene più assegnato un singolo, ma un vettore di valori, detto **vettore obiettivo**, ottenuto eseguendo tutte le funzioni obiettivo.



Algoritmi Genetici

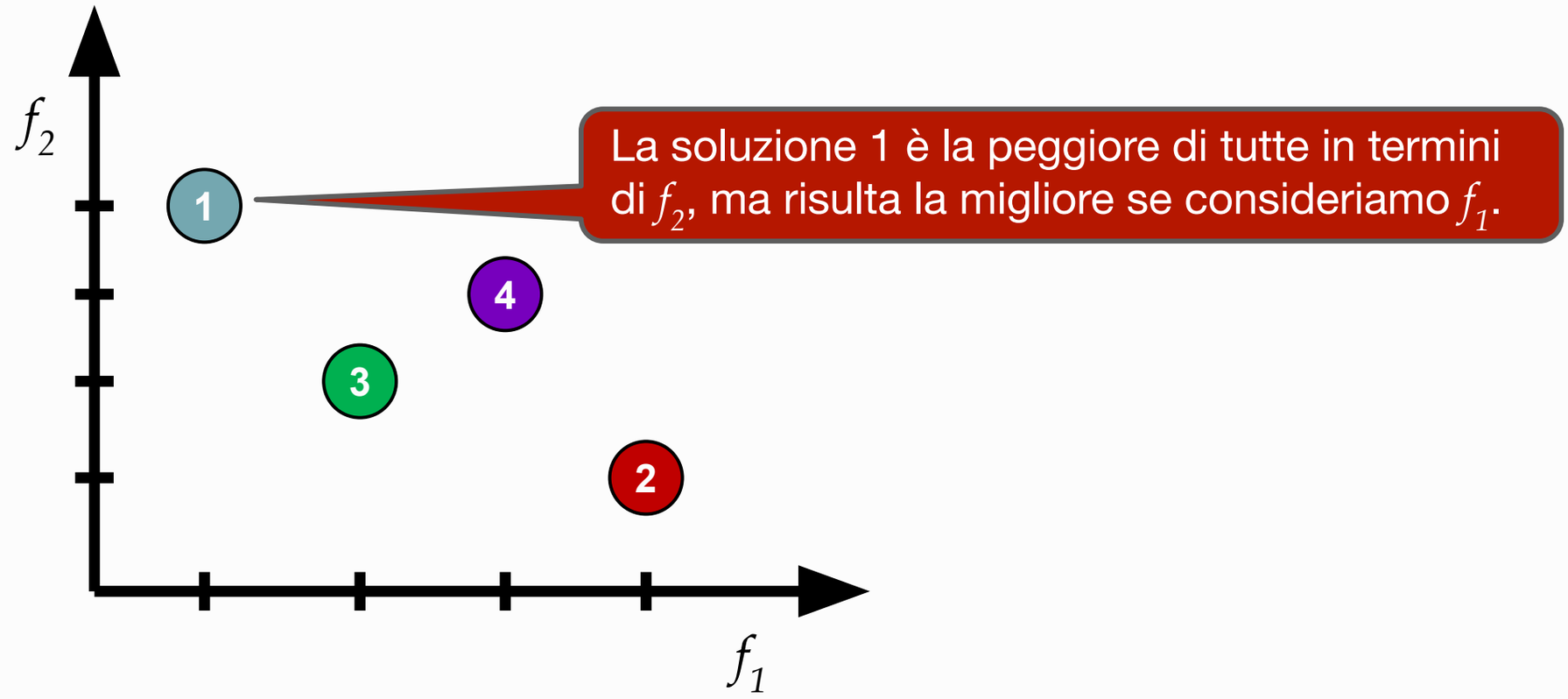
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

Ad ogni soluzione, quindi, non viene più assegnato un singolo, ma un vettore di valori, detto **vettore obiettivo**, ottenuto eseguendo tutte le funzioni obiettivo.



Algoritmi Genetici

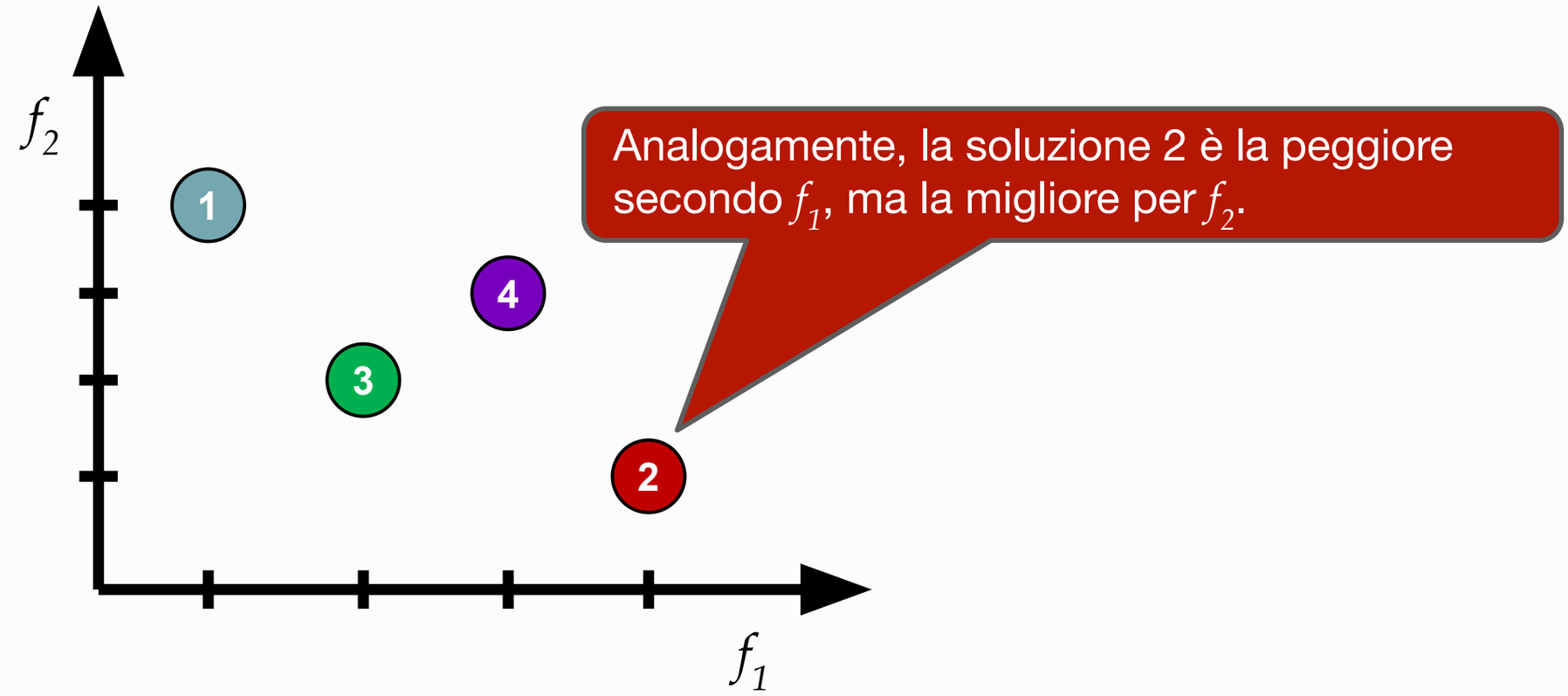
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

Ad ogni soluzione, quindi, non viene più assegnato un singolo, ma un vettore di valori, detto **vettore obiettivo**, ottenuto eseguendo tutte le funzioni obiettivo.



Algoritmi Genetici

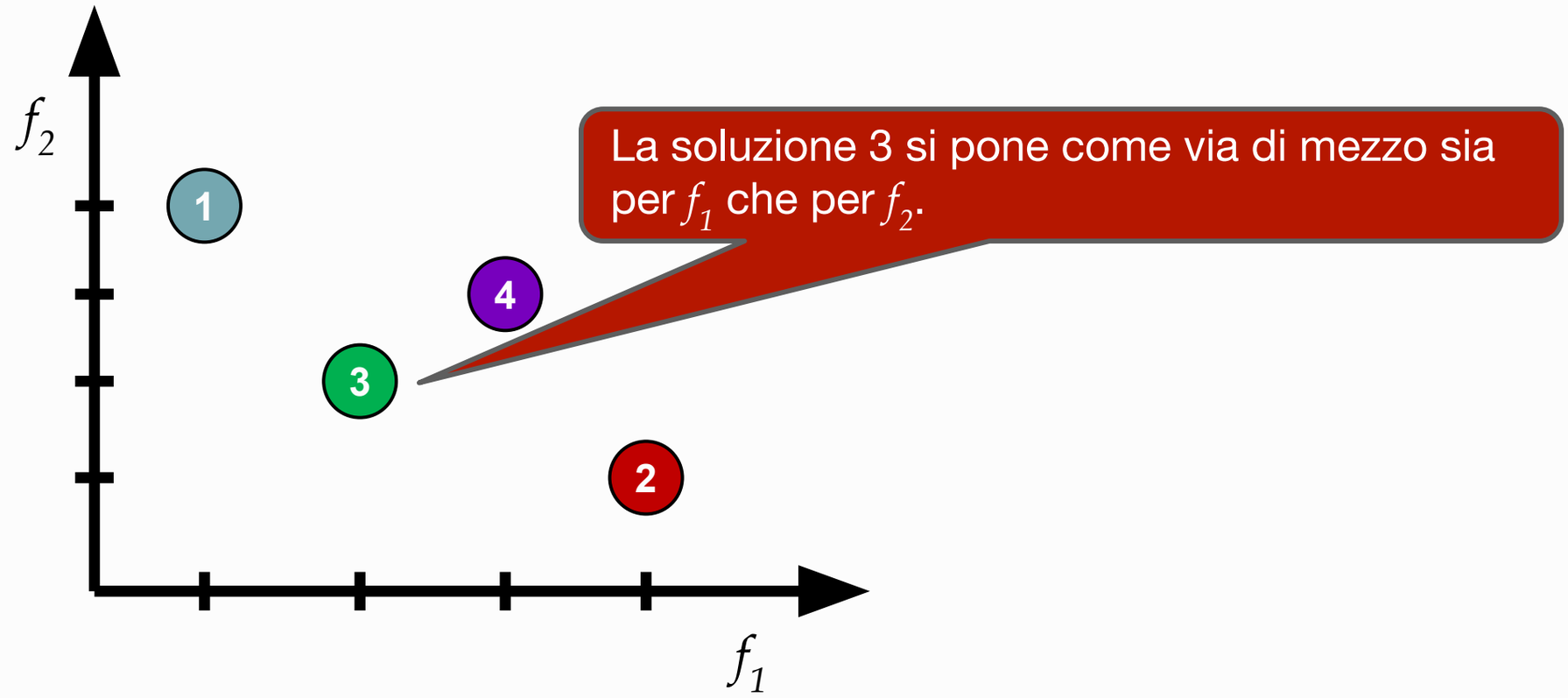
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

Ad ogni soluzione, quindi, non viene più assegnato un singolo, ma un vettore di valori, detto **vettore obiettivo**, ottenuto eseguendo tutte le funzioni obiettivo.



Algoritmi Genetici

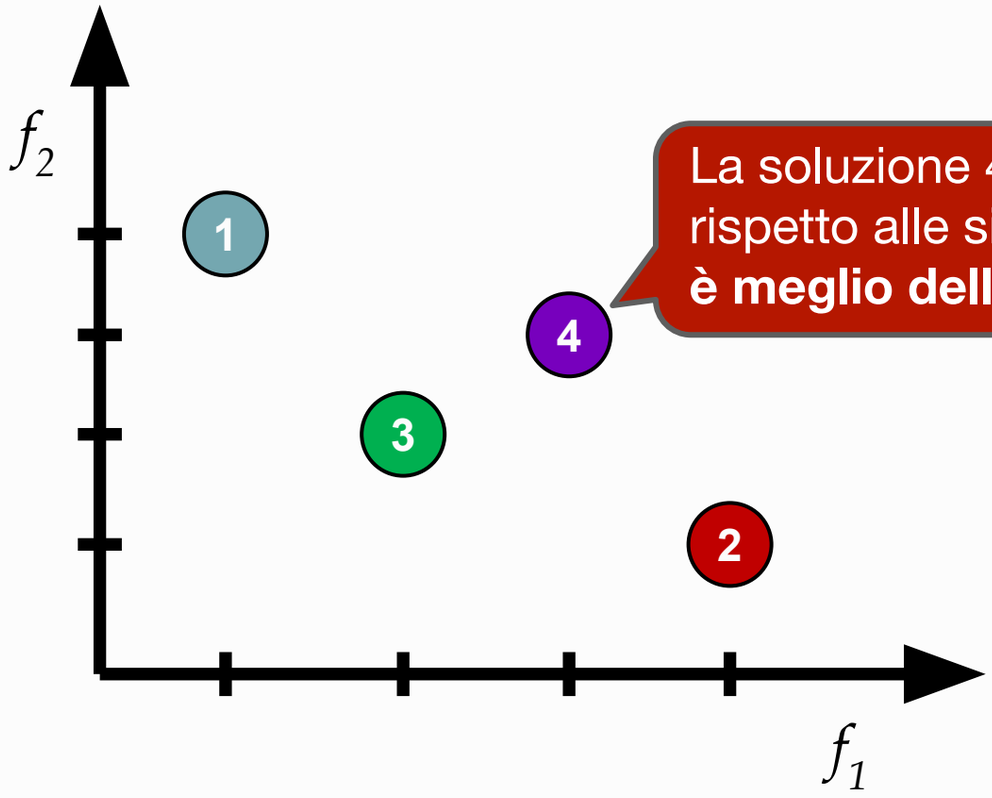
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

Ad ogni soluzione, quindi, non viene più assegnato un singolo, ma un vettore di valori, detto **vettore obiettivo**, ottenuto eseguendo tutte le funzioni obiettivo.



La soluzione 4, per quanto non sia la peggiore rispetto alle singole funzioni, sicuramente **non è meglio della soluzione 3**.

Algoritmi Genetici

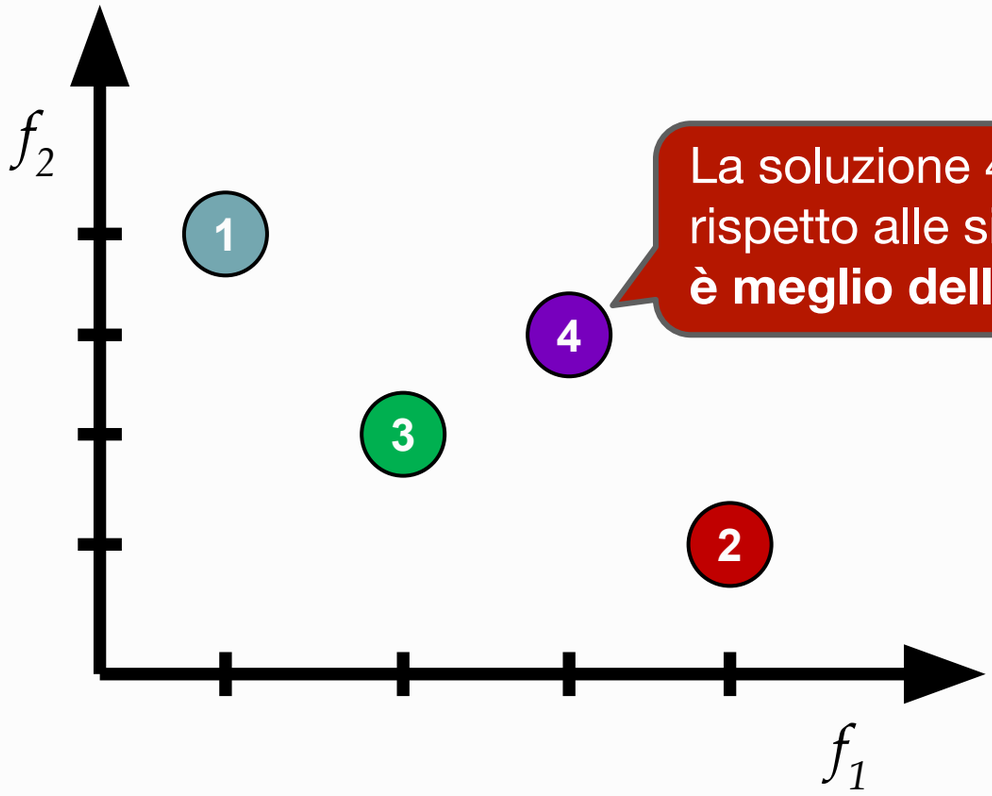
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

Ad ogni soluzione, quindi, non viene più assegnato un singolo, ma un vettore di valori, detto **vettore obiettivo**, ottenuto eseguendo tutte le funzioni obiettivo.



La soluzione 4, per quanto non sia la peggiore rispetto alle singole funzioni, sicuramente **non è meglio della soluzione 3**.

In questo scenario si dice che la Soluzione 3 **domina** la soluzione 4, poiché:

1. **Non è peggiore** rispetto a TUTTE le funzioni obiettivo;
2. **E' migliore** in ALMENO UNA funzione obiettivo.

Algoritmi Genetici

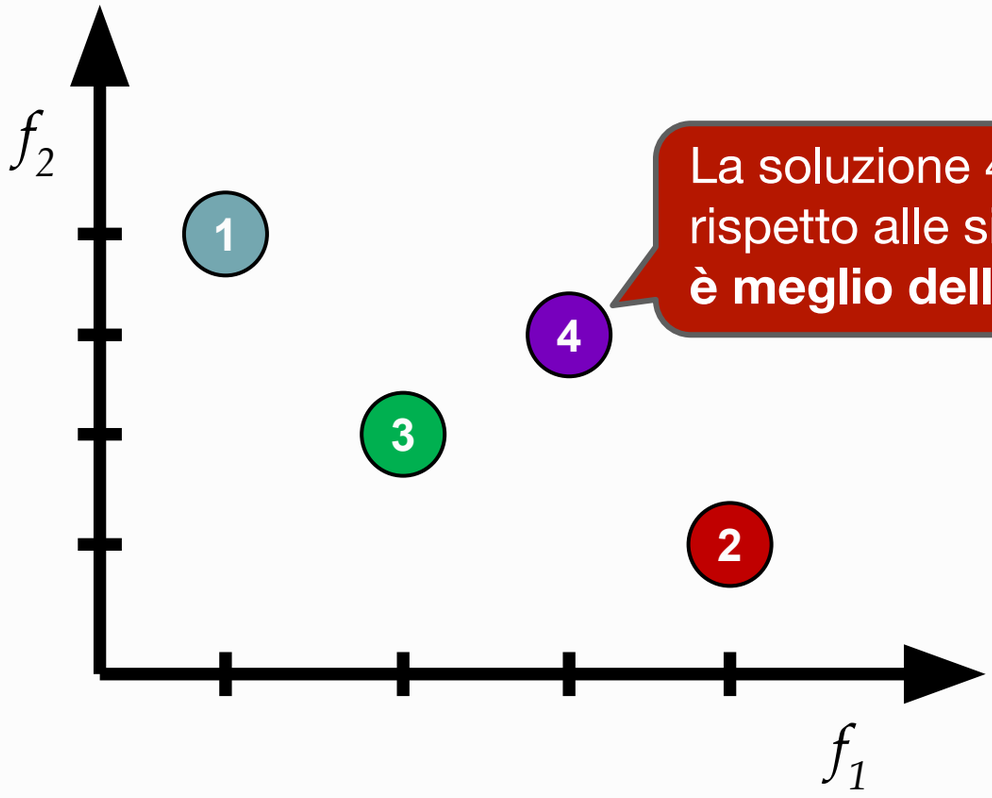
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

Ad ogni soluzione, quindi, non viene più assegnato un singolo, ma un vettore di valori, detto **vettore obiettivo**, ottenuto eseguendo tutte le funzioni obiettivo.



La soluzione 4, per quanto non sia la peggiore rispetto alle singole funzioni, sicuramente **non è meglio della soluzione 3**.

In formule: dato un problema di minimo con M funzioni obiettivo, una soluzione x **domina** una soluzione y (denotato con $x \preceq y$) se e solo se vale la seguente:

$$\forall i \in \{1, ..., M\}, f_i(x) \leq f_i(y) \; \textbf{AND} \; \exists j \in \{1, ..., M\}, f_j(x) < f_j(y)$$

Algoritmi Genetici

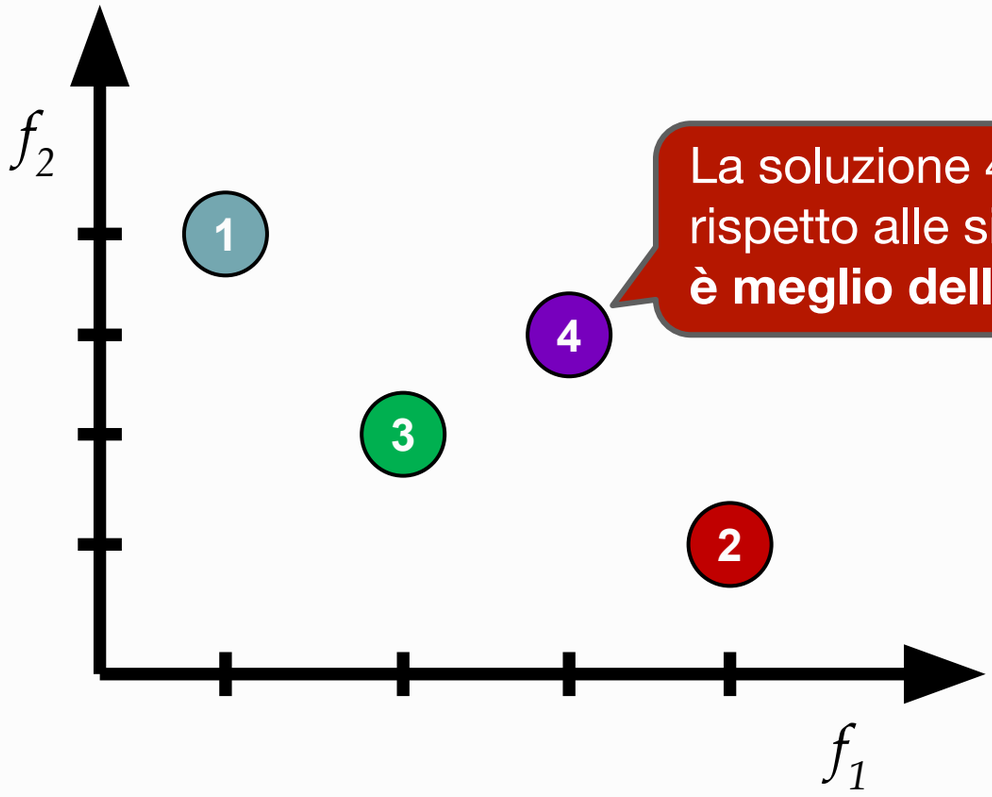
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

Ad ogni soluzione, quindi, non viene più assegnato un singolo, ma un vettore di valori, detto **vettore obiettivo**, ottenuto eseguendo tutte le funzioni obiettivo.



La soluzione 4, per quanto non sia la peggiore rispetto alle singole funzioni, sicuramente **non è meglio della soluzione 3**.

Questa definizione definisce una *relazione* con le seguenti proprietà:

- NON riflessiva (non è vero che $x \preceq x$)
- Asimmetrica (se $x \preceq y$, non è vero che $y \preceq x$)
- Transitiva (se $x \preceq y$ e $y \preceq z$, allora $x \preceq z$)

Ordine parziale stretto: possiamo confrontare soltanto alcune delle soluzioni.

Algoritmi Genetici

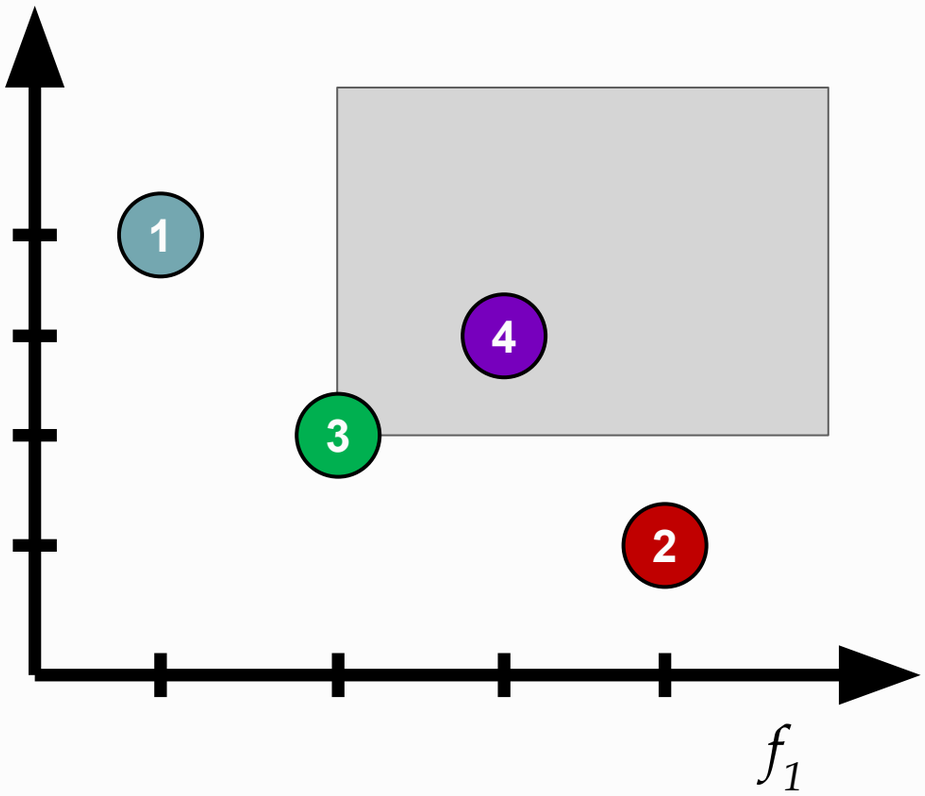
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

E' possibile identificare graficamente l'**insieme dominato** (*dominated set*) da una soluzione in spazi obiettivo bidimensionali.



Algoritmi Genetici

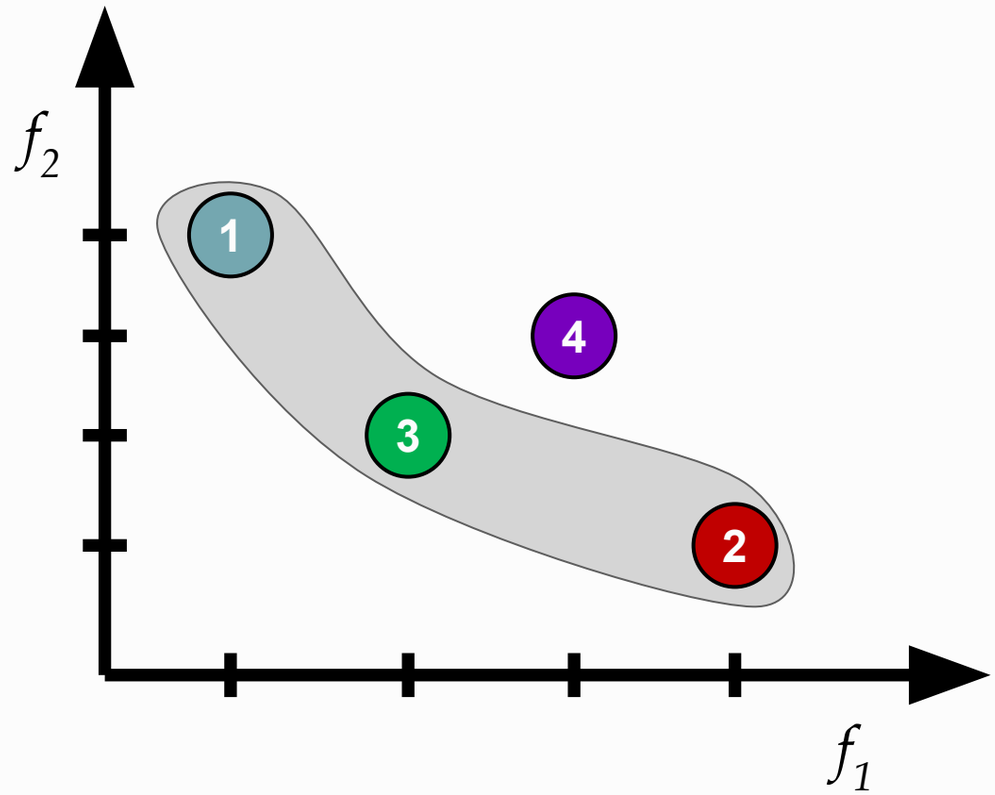
Problemi Multi-Obiettivo

Qual è il problema principale della modellazione ideale con M obiettivi contrastanti?

Ciascun obiettivo, preso singolarmente, ha sicuramente un suo punto di ottimo, ma se consideriamo tutti gli obiettivi è chiaro che il concetto di ottimo vada riconsiderato.

Consideriamo il seguente **spazio obiettivo** di due funzioni obiettivo in conflitto f_1 ed f_2, entrambe da minimizzare, e consideriamo l'esistenza soltanto di quattro soluzioni.

E' possibile identificare graficamente l'**insieme dominato** (*dominated set*) da una soluzione in spazi obiettivo bidimensionali.



Le soluzioni che non compaiono in alcun insieme dominato formano l'**insieme delle soluzioni non-dominate** (*non-dominated set*). Questo insieme, a livello globale, è anche noto come **Fronte di Pareto**, che contiene tutte le soluzioni ottimali in trade-off. Le soluzioni nel Fronte sono dette **ottime secondo Pareto** (*Pareto-optimal solutions*).

Algoritmi Genetici

Problemi Multi-Obiettivo

A livello teorico lo scopo finale di un ottimizzatore multi-obiettivo sarebbe quello di trovare il Fronte di Pareto e restituire tutte le soluzioni in esso contenuto.
Un'ottimizzazione del genere è anche "banale" da realizzare, infatti bisogna:

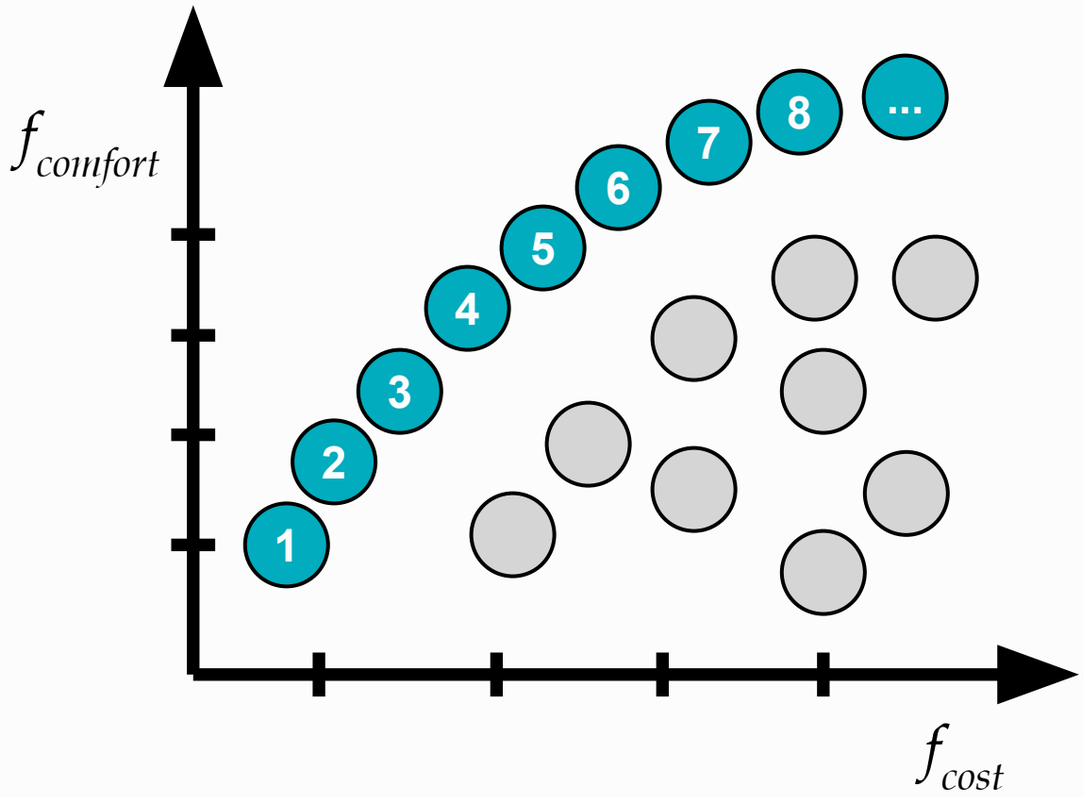
1. valutare tutte le soluzioni rispetto a tutte le M funzioni obiettivo;
2. confrontare ogni soluzione con tutte le altre (stabilendo i link di dominanza);
3. le soluzioni mai dominate formano il Fronte di Pareto.

Questa soluzione presenta due problemi. Il primo riguarda il **carico computazionale**: è impensabile valutare tutte le soluzioni nello spazio di ricerca (è una *ricerca esaustiva*). Il secondo è il seguente: "*abbiamo davvero bisogno di TUTTO il Fronte di Pareto?*"

Nell'esempio precedente il Fronte di Pareto era estremamente piccolo (3 soluzioni), il che è totalmente accettabile, ma nei problemi del mondo reale abbiamo **spazi obiettivo più grandi** (più di due obiettivi) ed uno **spazio di ricerca MOLTO più grande** (tante variabili input). Questo implica un numero minore di relazioni di dominanza, allargando a dismisura il Fronte di Pareto!

Algoritmi Genetici

Problemi Multi-Obiettivo



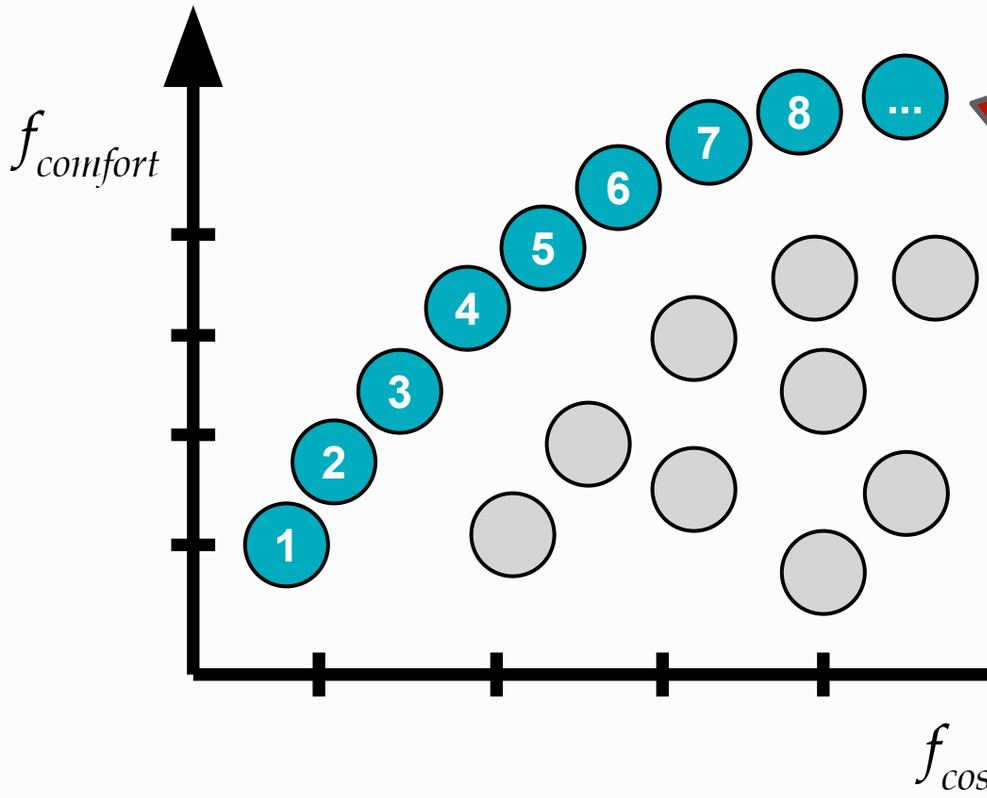
Ritorniamo all'esempio delle auto, dove vogliamo massimizzare il comfort e minimizzare il costo. Immaginiamo che l'algoritmo ci restituisca un insieme di 300 auto, tutte appartenenti al Fronte di Pareto. Qual è il problema?

Come potrei decidere quale auto farà al caso nostro? Il mio processo di decision-making ne risentirebbe, rendendo del tutto **inutile** aver fatto l'ottimizzazione.

Quindi ci chiediamo: "*siamo davvero interessati a TUTTE le soluzioni in trade-off?"*. Non c'è un modo per avere soltanto le *soluzioni principali*, magari quelle più "interessanti"?

Algoritmi Genetici

Problemi Multi-Obiettivo



Ad esempio, le soluzione 4 e 5 sono molto simili tra loro, quindi ottenerle entrambe non ci farebbe molta differenza: hanno poco trade-off. Discorso diverso per le soluzioni 1 ed 8, che hanno un trade-off molto forte, e quindi sono molto più interessanti.

In sostanza, ci piacciono le soluzioni **diversificate**, quindi vorremmo un **sottoinsieme del Fronte di Pareto con soluzioni ad alto trade-off**.

Ritorniamo all'esempio delle auto, dove vogliamo massimizzare il comfort e minimizzare il costo. Immaginiamo che l'algoritmo ci restituisca un insieme di 300 auto, tutte appartenenti al Fronte di Pareto. Qual è il problema?

Come potrei decidere quale auto farà al caso nostro? Il mio processo di decision-making ne risentirebbe, rendendo del tutto **inutile** aver fatto l'ottimizzazione.

Quindi ci chiediamo: "*siamo davvero interessati a TUTTE le soluzioni in trade-off?*". Non c'è un modo per avere soltanto le *soluzioni principali*, magari quelle più "interessanti"?

Algoritmi Genetici

Problemi Multi-Obiettivo

Ci occorre, quindi, una strategia in grado di:

- ottenere soluzioni ben **diversificate**, pur non necessariamente Pareto-optimal;
- evitare la ricerca esaustiva per avere soluzioni in **tempi accettabili.**

Hill Climbing, così come tutti gli altri algoritmi di ricerca locale “pura”, sono in grado di restituire una singola soluzione sub-ottimale, non un insieme.

Sarebbe necessario, quindi, adottare una variante di Hill Climbing, magari eseguendolo più volte da diversi punti dello spazio di ricerca (sulla falsariga della variante con “riavvio casuale”). I risultati di tutte le esecuzioni formerebbero un **insieme di soluzioni vicino al Fronte di Pareto**.

E’ un’idea molto semplice, ma non è molto efficiente (per via delle molteplici esecuzioni) e comunque non ci garantisce molta diversità tra le soluzioni identificate.

Servirebbe un algoritmo che *naturalmente* migliora ad ogni iterazione un insieme di soluzioni candidate invece che una soltanto: un **algoritmo genetico**.

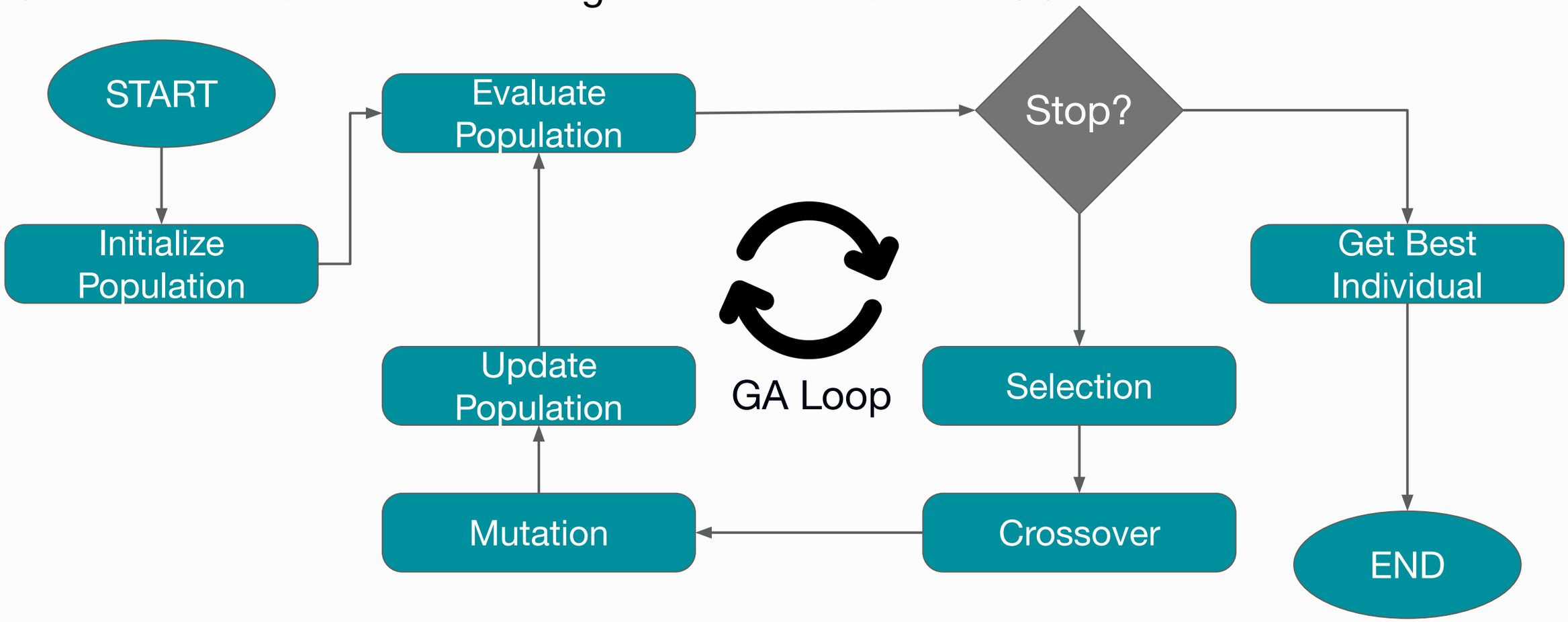
Un GA, così come tutte le meta-euristiche population-based, risponde perfettamente alle nostre esigenze, poiché nella sua ultima iterazione dispone già di un insieme di soluzioni **sub-ottimali** (vicine al Fronte) e **ben diversificate.**

Quali accorgimenti bisogna adottare per rendere i GA utilizzabili nella MOO?

Algoritmi Genetici

GA per MOO

Consideriamo il funzionamento generale di un classico SGA.

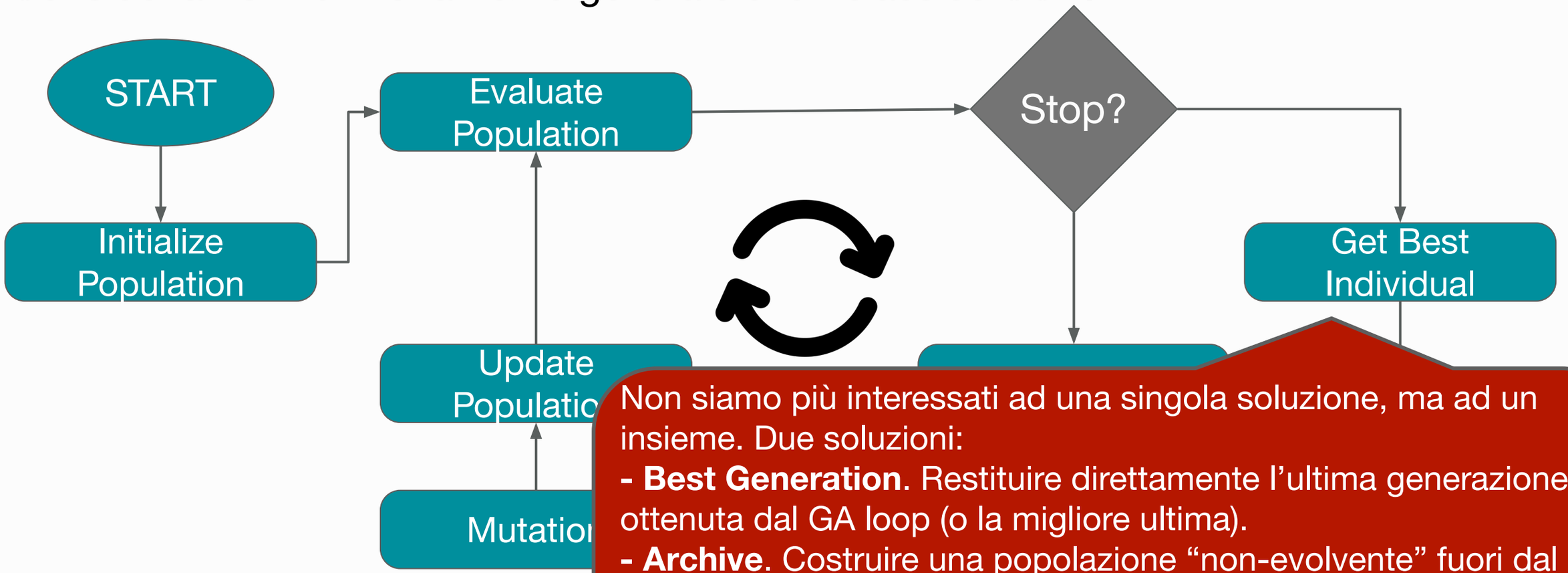


Quali componenti bisogna modificare?

Algoritmi Genetici

GA per MOO

Consideriamo il funzionamento generale di un classico SGA.



Quali componenti bisogna modif

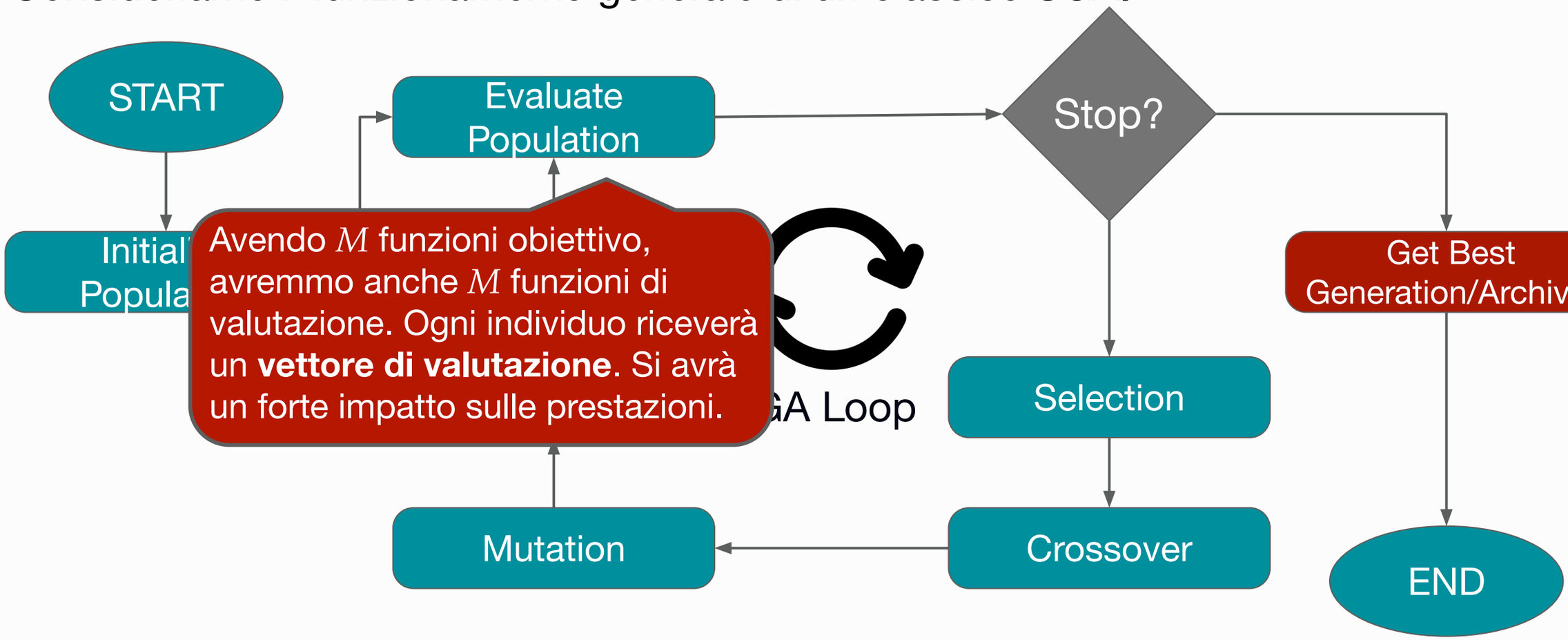
Non siamo più interessati ad una singola soluzione, ma ad un insieme. Due soluzioni:

- **Best Generation**. Restituire direttamente l'ultima generazione ottenuta dal GA loop (o la migliore ultima).
- **Archive**. Costruire una popolazione "non-evolvente" fuori dal loop che contiene i migliori individui di ciascuna generazione. Se il miglior individuo ottenuto ad una certa iterazione domina un altro già presente nell'archivio, allora prende il suo posto, altrimenti lo si aggiunge. Al termine del GA loop, si restituisce l'archivio.

Algoritmi Genetici

GA per MOO

Consideriamo il funzionamento generale di un classico SGA.

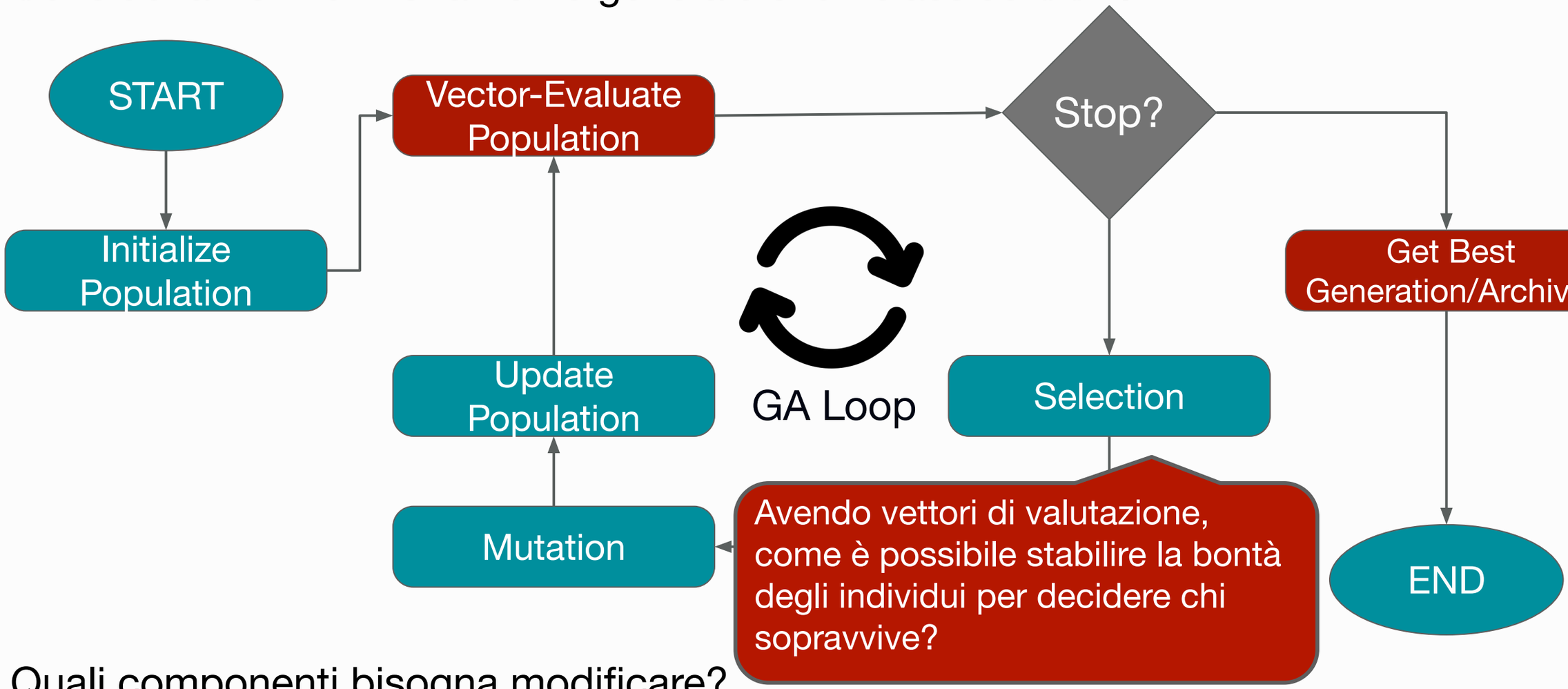


Quali componenti bisogna modificare?

Algoritmi Genetici

GA per MOO

Consideriamo il funzionamento generale di un classico SGA.

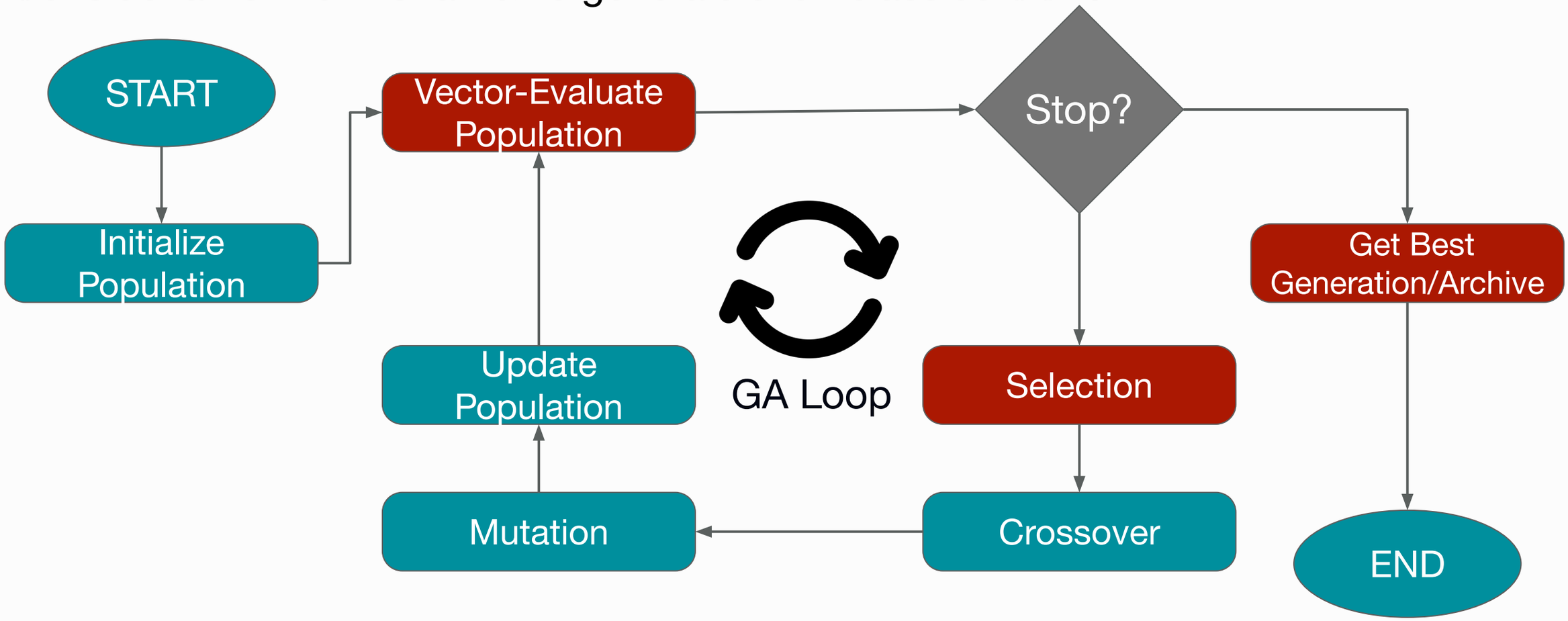


Quali componenti bisogna modificare?

Algoritmi Genetici

GA per MOO

Consideriamo il funzionamento generale di un classico SGA.



Quali componenti bisogna modificare?

La selezione risulta il nodo più difficile da sciogliere. Infatti, non possiamo applicare i classici operatori di selezione senza opportuni accorgimenti. In ogni caso, ci occorre necessariamente un **metodo di ordinamento degli individui** all'interno della popolazione, visto che non è facile confrontare dei vettori tra loro. Negli anni sono state proposte diverse soluzioni più o meno valide. Vediamo le tre più importanti.

Algoritmi Genetici

GA per MOO - Ordinamento della Popolazione

Soluzione di VEGA

Proposta da Schaffer nel 1984 nell'ambito dell'algoritmo VEGA (*Vector-Evaluated GA*), probabilmente il primo GA usato per MOO.

Dato un GA con M funzioni di valutazione, si divide la popolazione casualmente in M sotto-popolazioni. Ciascuna utilizzerà una delle M funzioni di valutazione per ottenere la funzione di fitness e compiere la selezione internamente alla sotto-popolazione.

Molto semplice, ma tende a preferire troppo le soluzioni ottimali rispetto alle singole funzioni obiettivo, allontanandosi troppo dal fronte di Pareto. In altre parole, è vero che si ottiene molta diversità, ma non si considerano le soluzioni in trade-off.

Soluzione di MOGA

Proposta da Fonseca e Flemming nel 1993 nell'ambito dell'algoritmo MOGA (*Multi-Objective GA*), e fa uso delle relazioni di dominanza.

Ad ogni individuo x si assegna un rango pari al "numero di individui che dominano x" + 1. La funzione di fitness (che omettiamo perché poco intuitiva) va a mappare questi ranghi in una probabilità di sopravvivenza.

Questo implica che: le soluzioni nel fronte di Pareto avranno tutte rango 1. Non è detto (anzi, è quasi sicuro di no) che saranno assegnati tutti i rangi tra 1 e $|P|$.

GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

Proposta da Deb nell'ambito dell'algoritmo NSGA (*Non-Dominated Sorting Algorithm*) e in tutte le sue varianti. Si basa sul concetto di non-dominated set, e consiste nel creare un **ordinamento di tutti i non-dominated set** in diversi *livelli di non-dominanza* tramite una procedura iterativa, in una maniera simile al ranking applicato in MOGA.

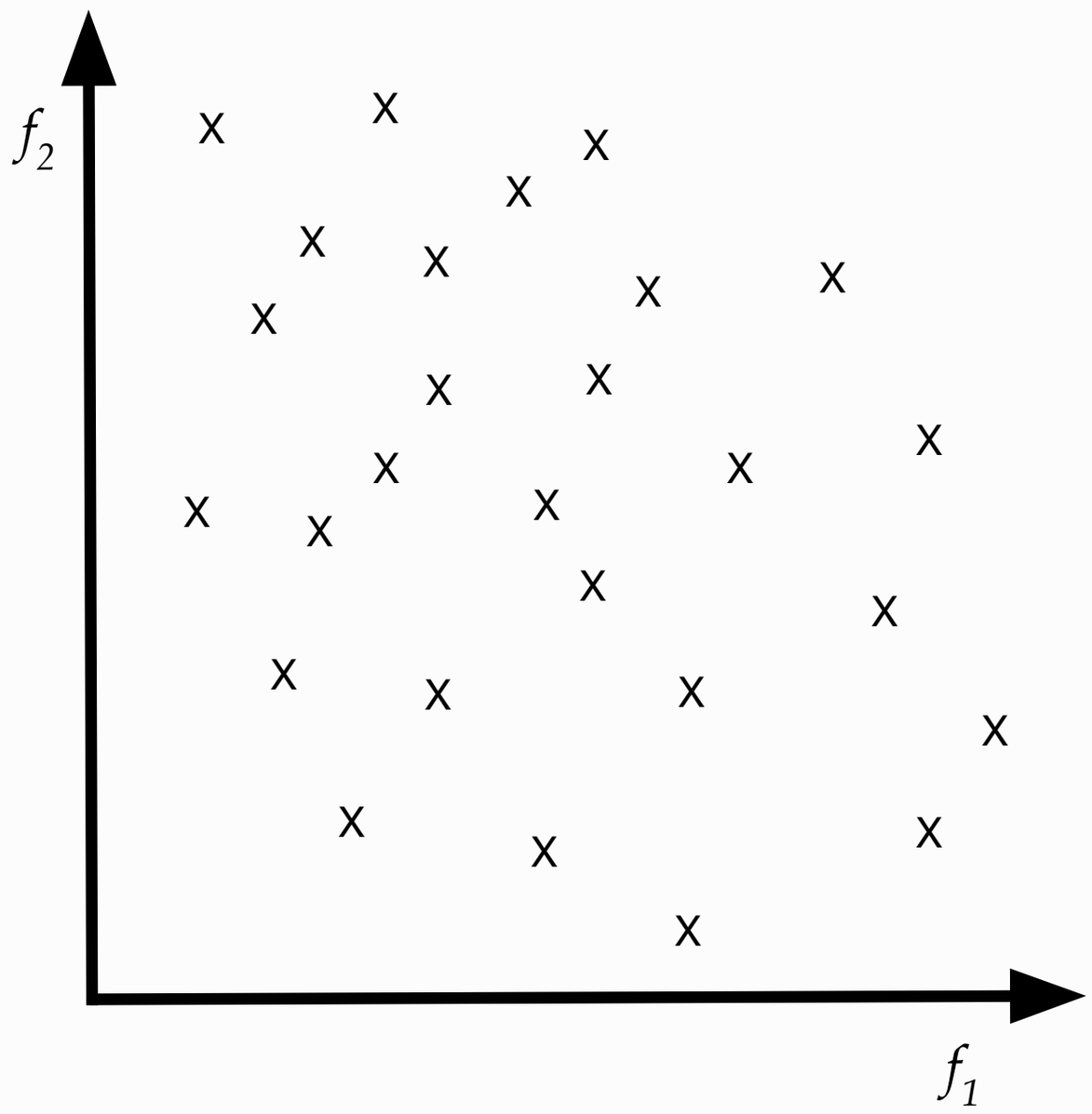
L'algoritmo di ordinamento compie i seguenti passi:

1. si valuta ogni individuo nella popolazione (ciascuno avrà un vettore di valutazione;
2. si inizializza il **contatore di livello**: $j = 1$;
3. si confronta tra loro tutti gli individui, allo scopo di **trovare gli individui non dominati**, aggiunti all'insieme P_j;
4. si memorizza all'interno degli individui il valore j;
5. si rimuove dalla popolazione P tutti gli individui presenti in P_j ed incrementa j di 1;
6. si va al punto 3 se la popolazione P non è vuota;
7. si restituiscono tutti gli insiemi P_j.

GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

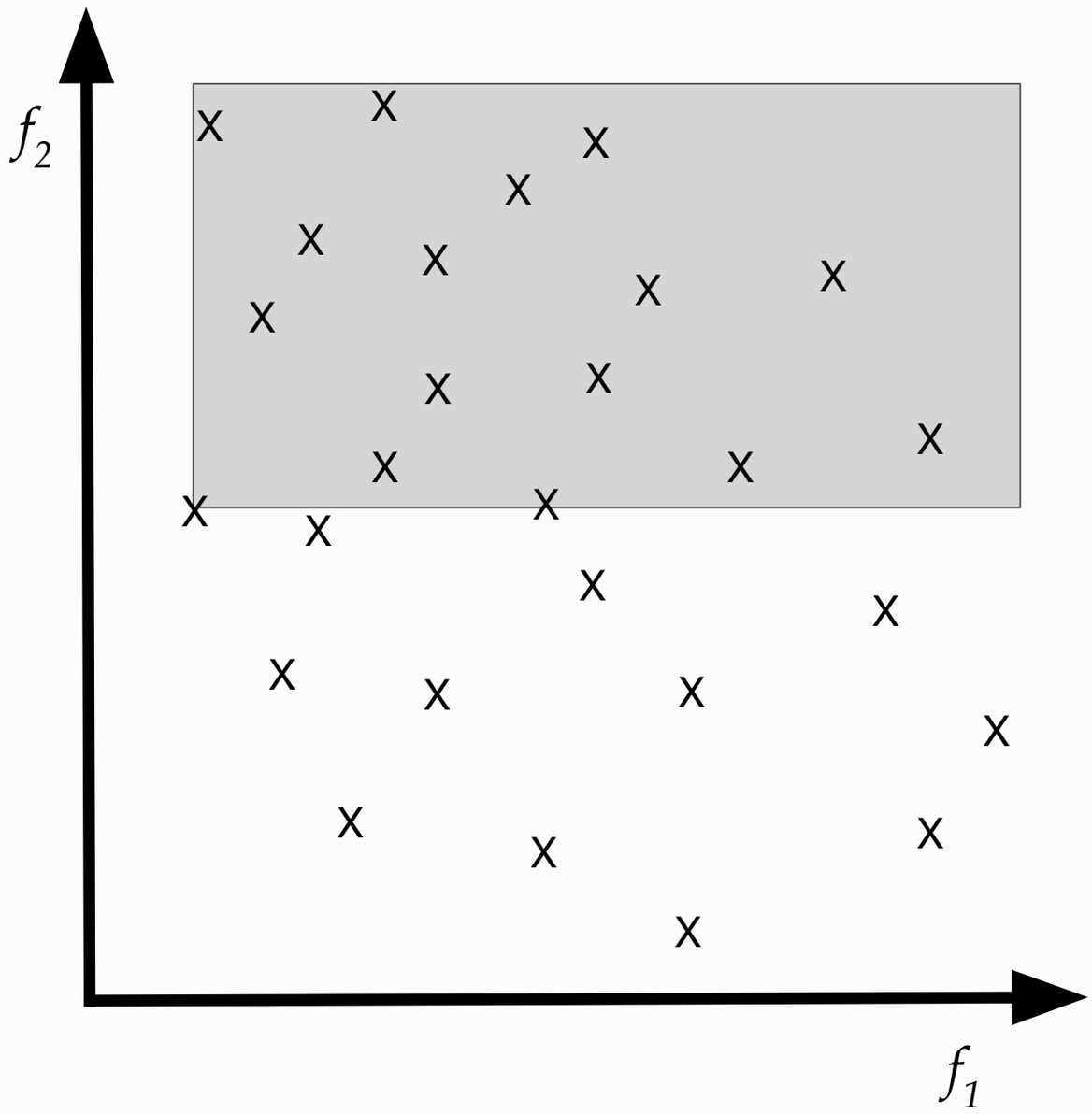
Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:



GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

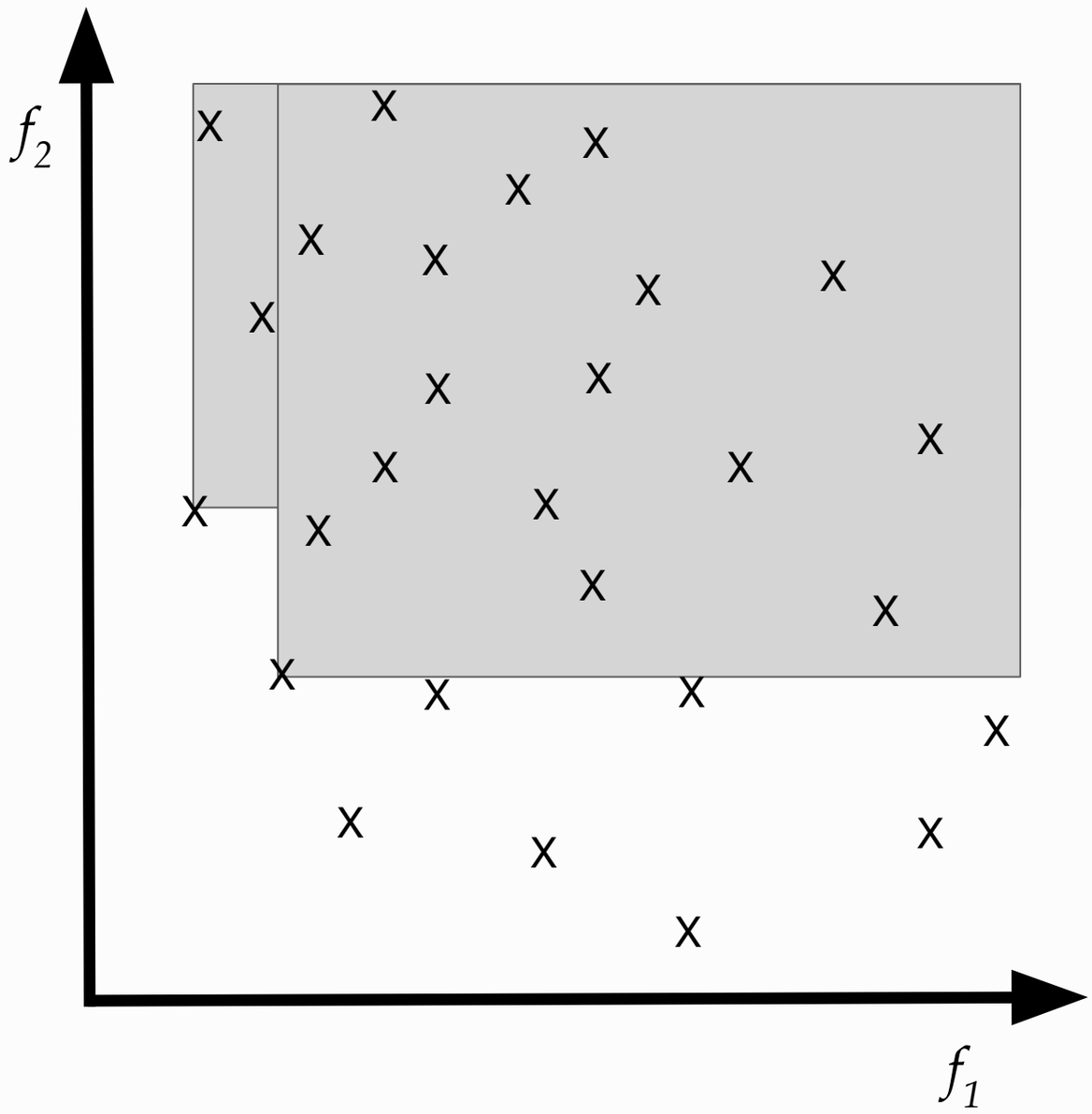
Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:



GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

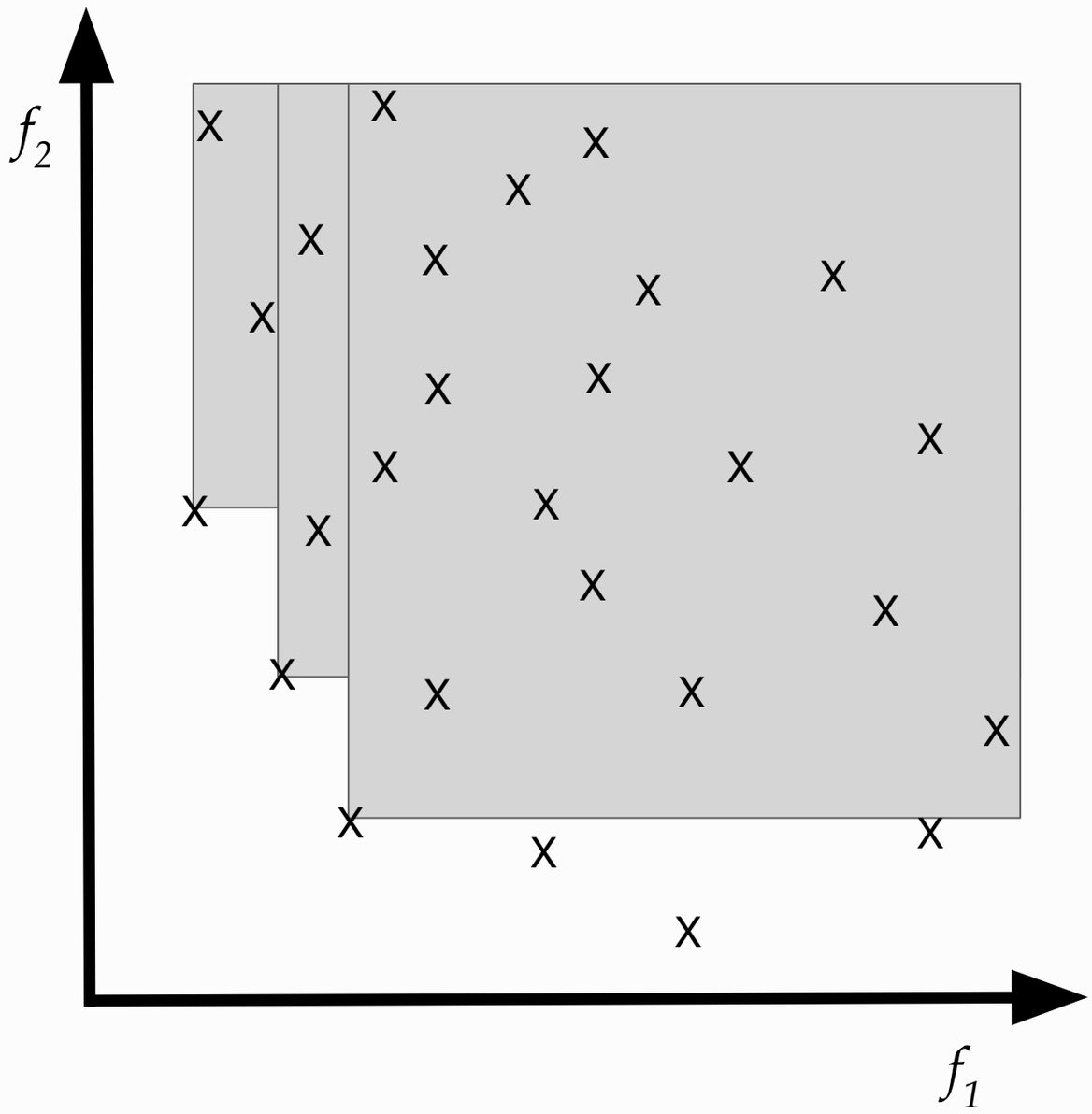
Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:



GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

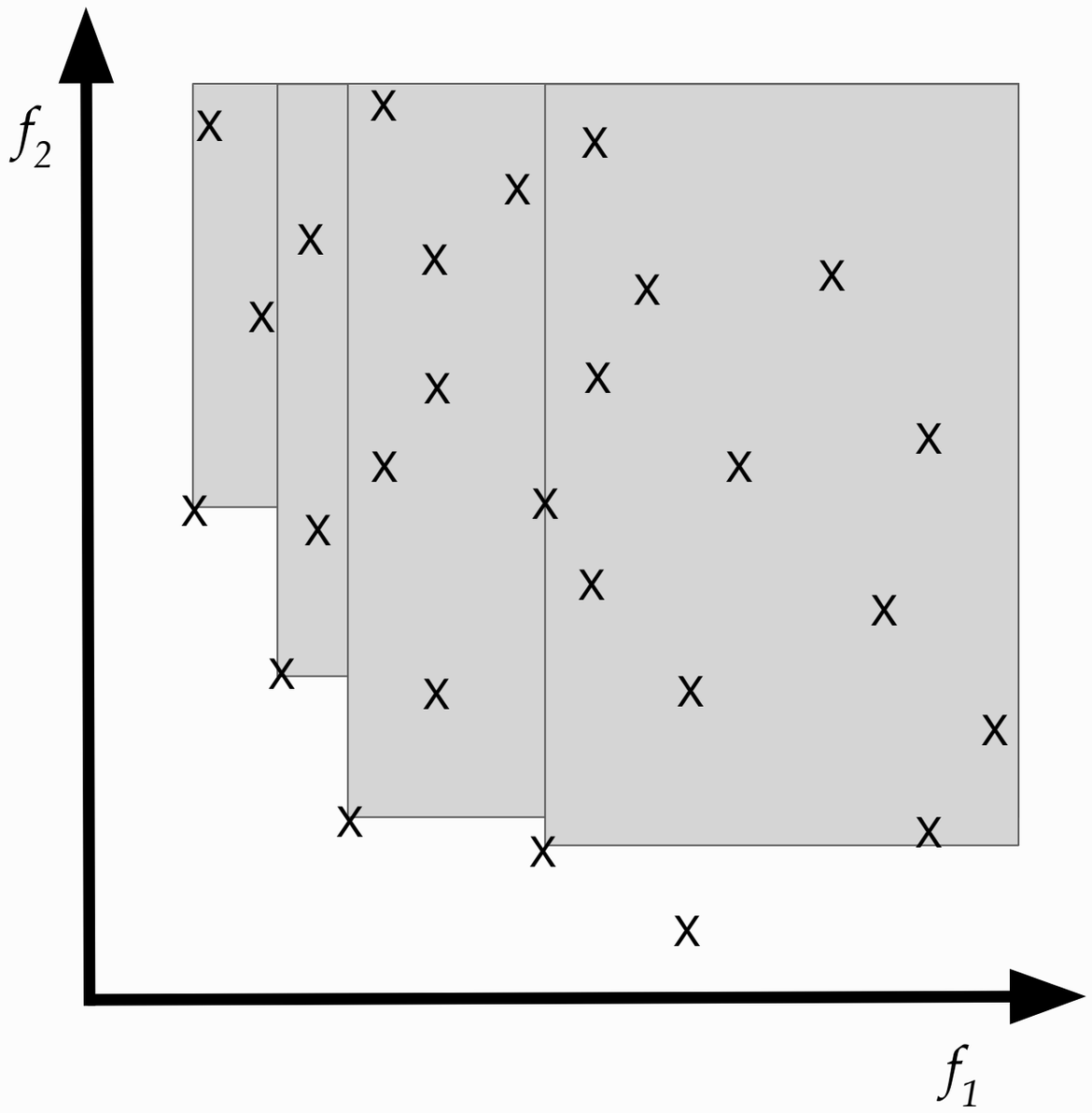
Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:



GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

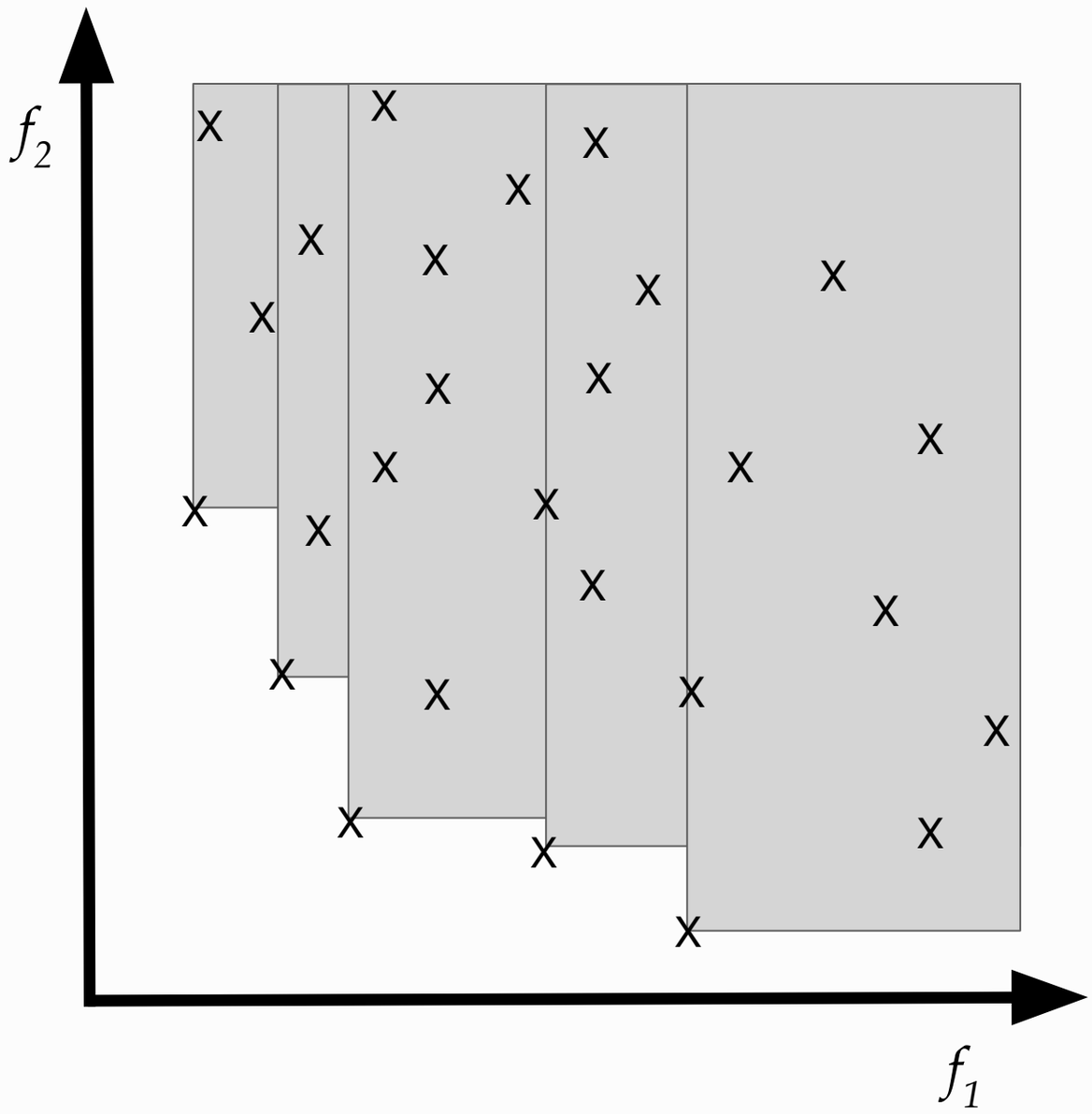
Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:



GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:



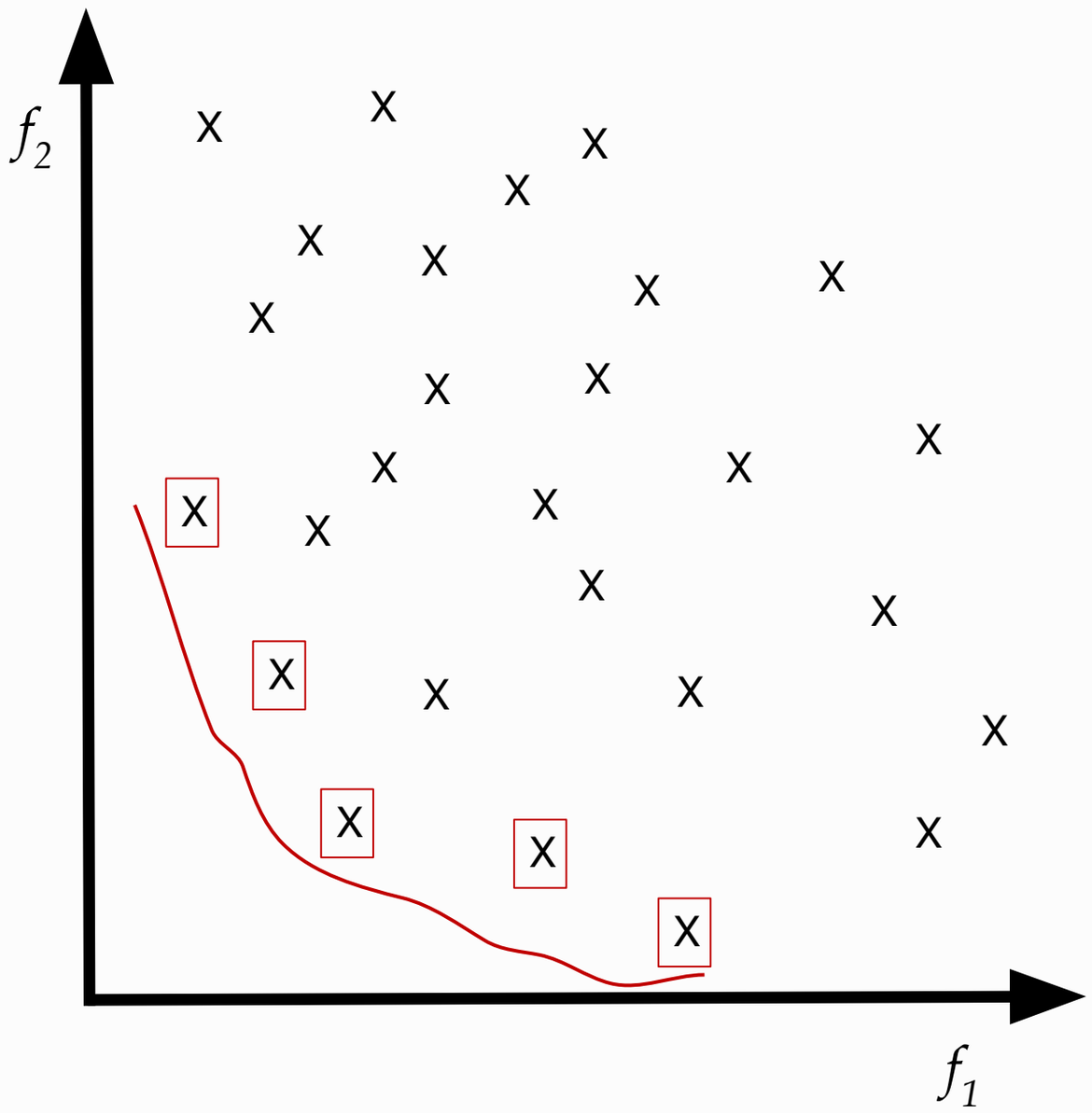
Algoritmi Genetici

GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:

Fronte 1:



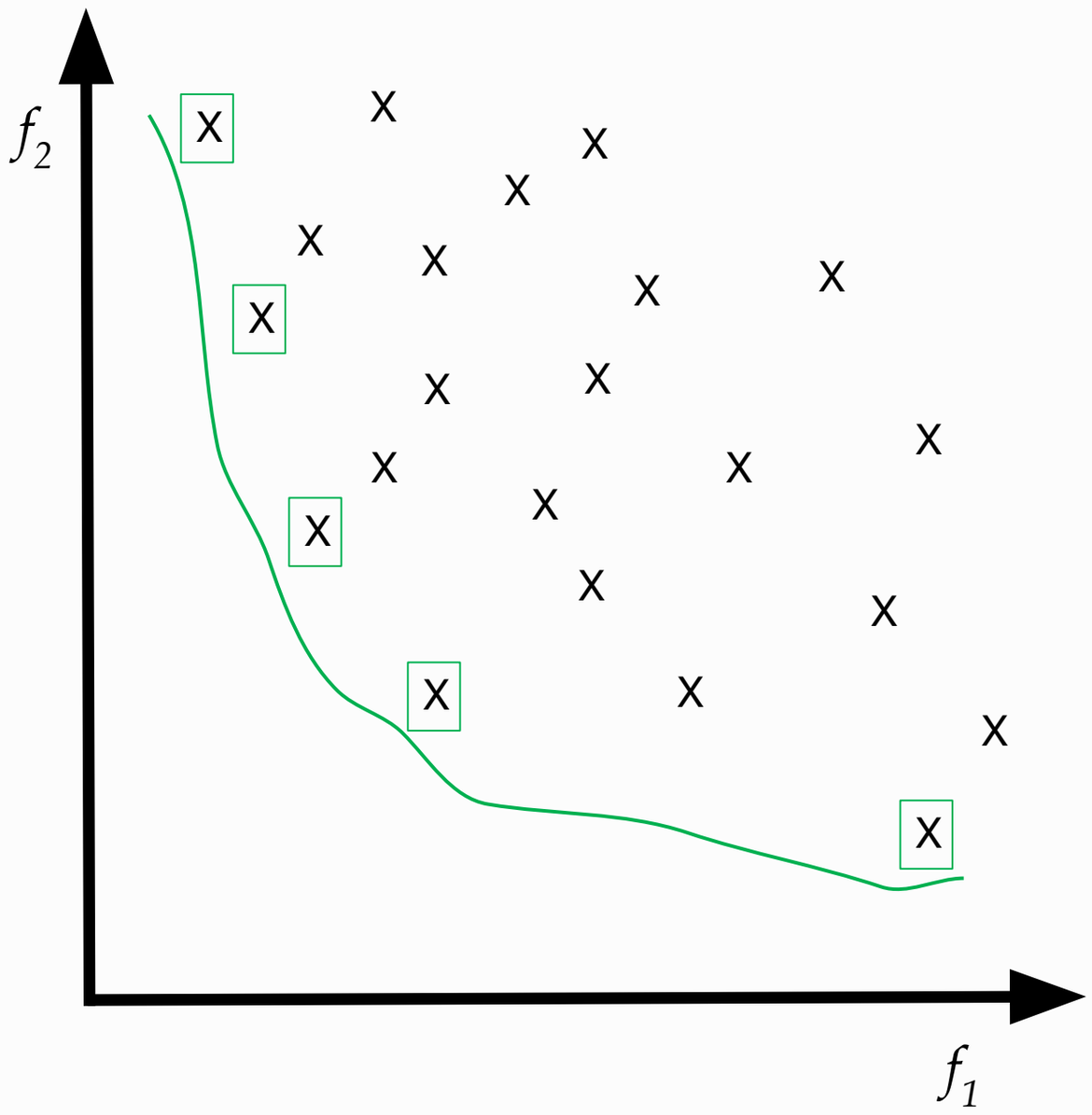
GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:

Fronte 1:

Fronte 2:

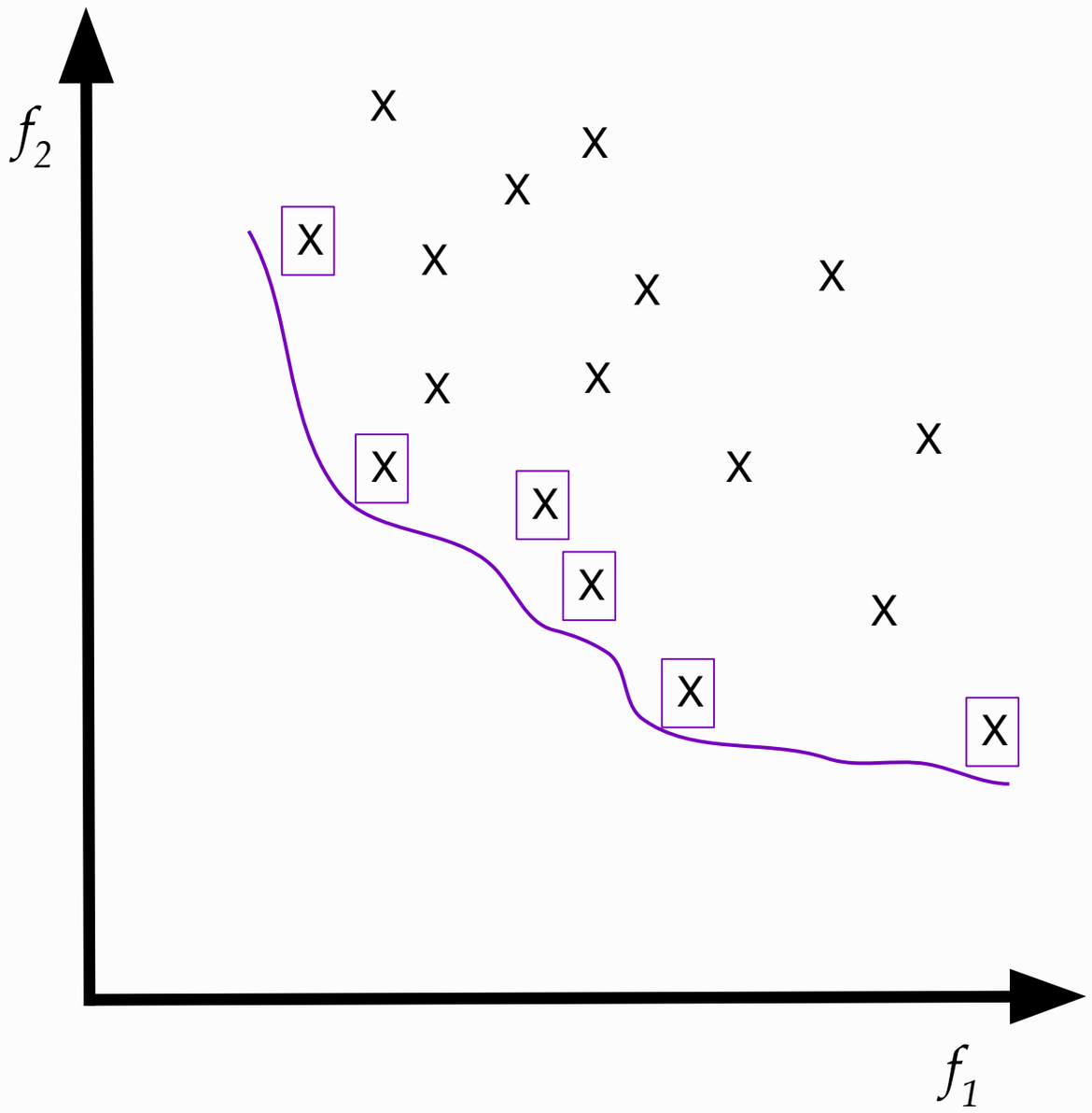


GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:

Fronte 1:
Fronte 2:
Fronte 3:



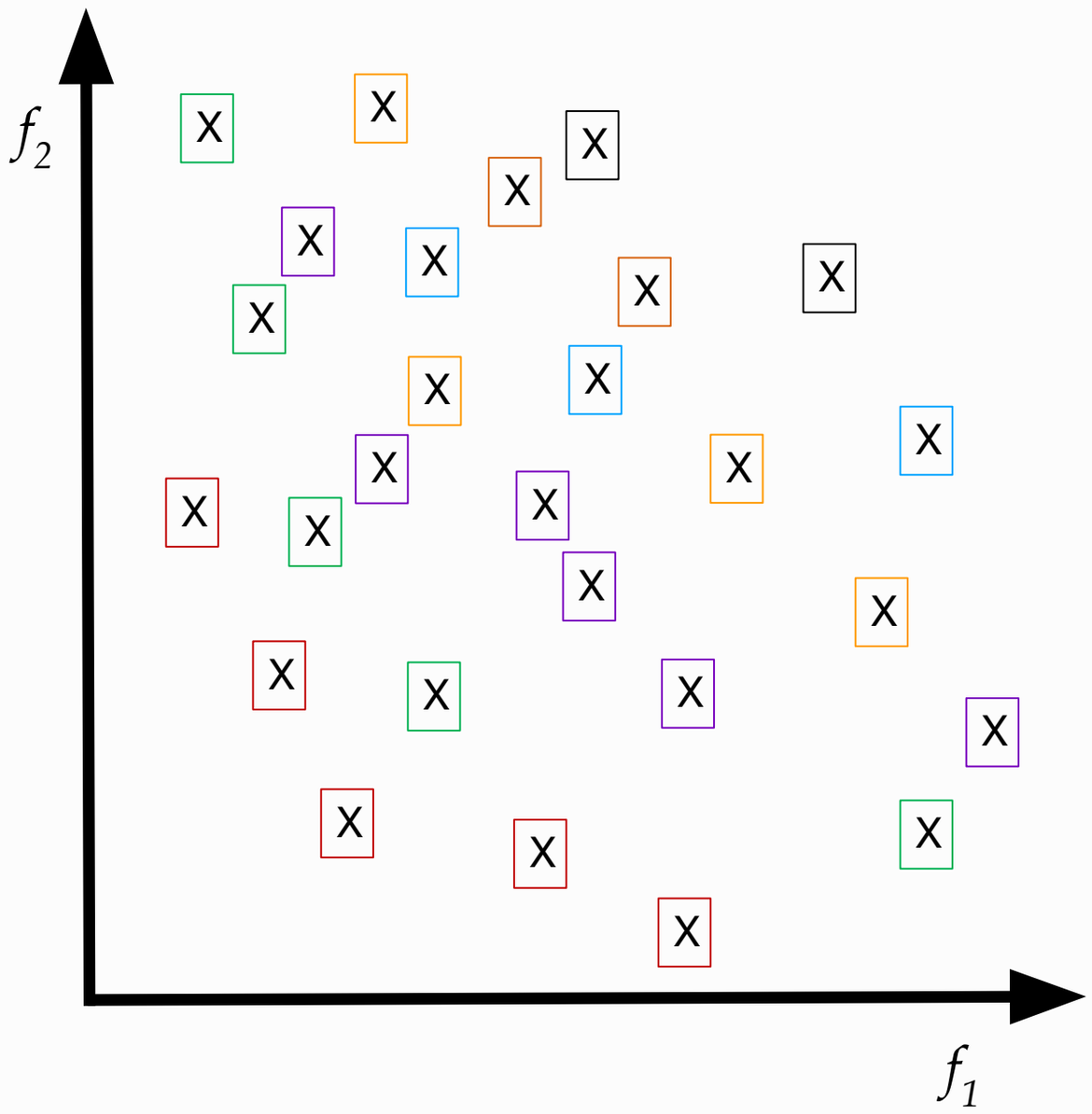
Algoritmi Genetici

GA per MOO - Ordinamento della Popolazione

Soluzione di NSGA

Vediamo una demo in uno spazio obiettivo bi-dimensionale da minimizzare:

Fronte 1:
Fronte 2:
Fronte 3:
Fronte 4:
Fronte 5:
Fronte 6:
Fronte 7:



GA per MOO - Selezione

Soluzione di NSGA

Sulla base del fronte di appartenenza possiamo gettare le basi del nostro algoritmo di selezione. Sicuramente possiamo già dire che tra due individui si **favorisce quello di livello minore** poiché è chiaramente più vicino al vero Fronte di Pareto.

Ma come ci comportiamo di fronte a *due individui appartenenti allo stesso livello*?

Soluzione banale. Si seleziona casualmente un fronte con probabilità analoga alla Rank Selection ($prob(l) = (numeroLivelli - l + 1) / \Sigma\, l$), per poi selezionare in maniera uniforme un individuo all'interno del fronte selezionato. In altre parole, **nessun individuo all'interno del medesimo fronte è migliore di un altro**.

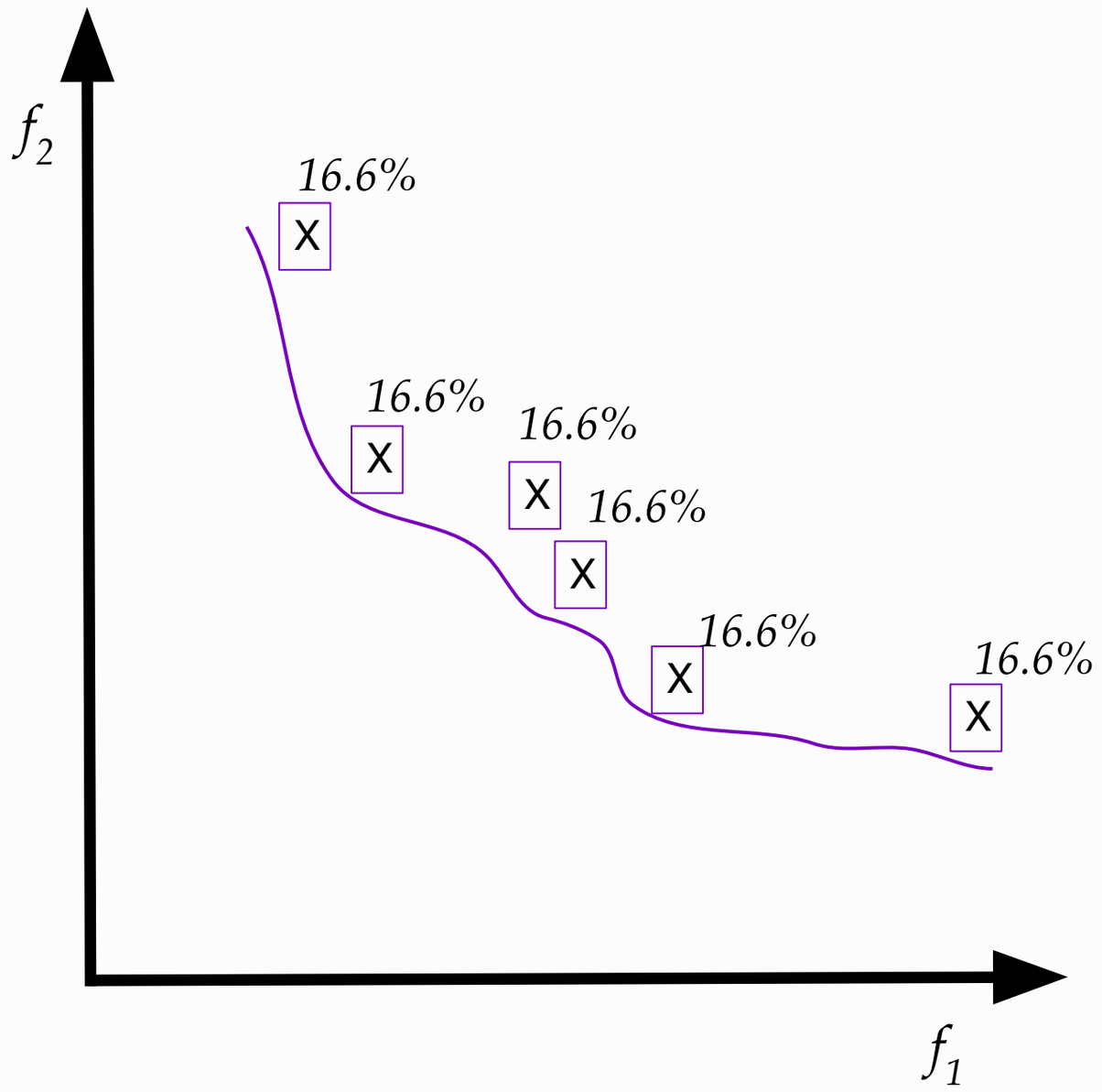
Che problemi dà questa soluzione?

La probabilità di selezionare due o più individui con trade-off simile è molto più alta rispetto alla probabilità di selezionare individui con trade-off diversi.

GA per MOO - Selezione

Soluzione di NSGA

Ritorniamo all'esempio precedente, ed immaginiamo di selezionare il Fronte 3. Ciascun individuo ha circa il 16.6% di probabilità di essere selezionato.



Algoritmi Genetici

GA per MOO - Selezione

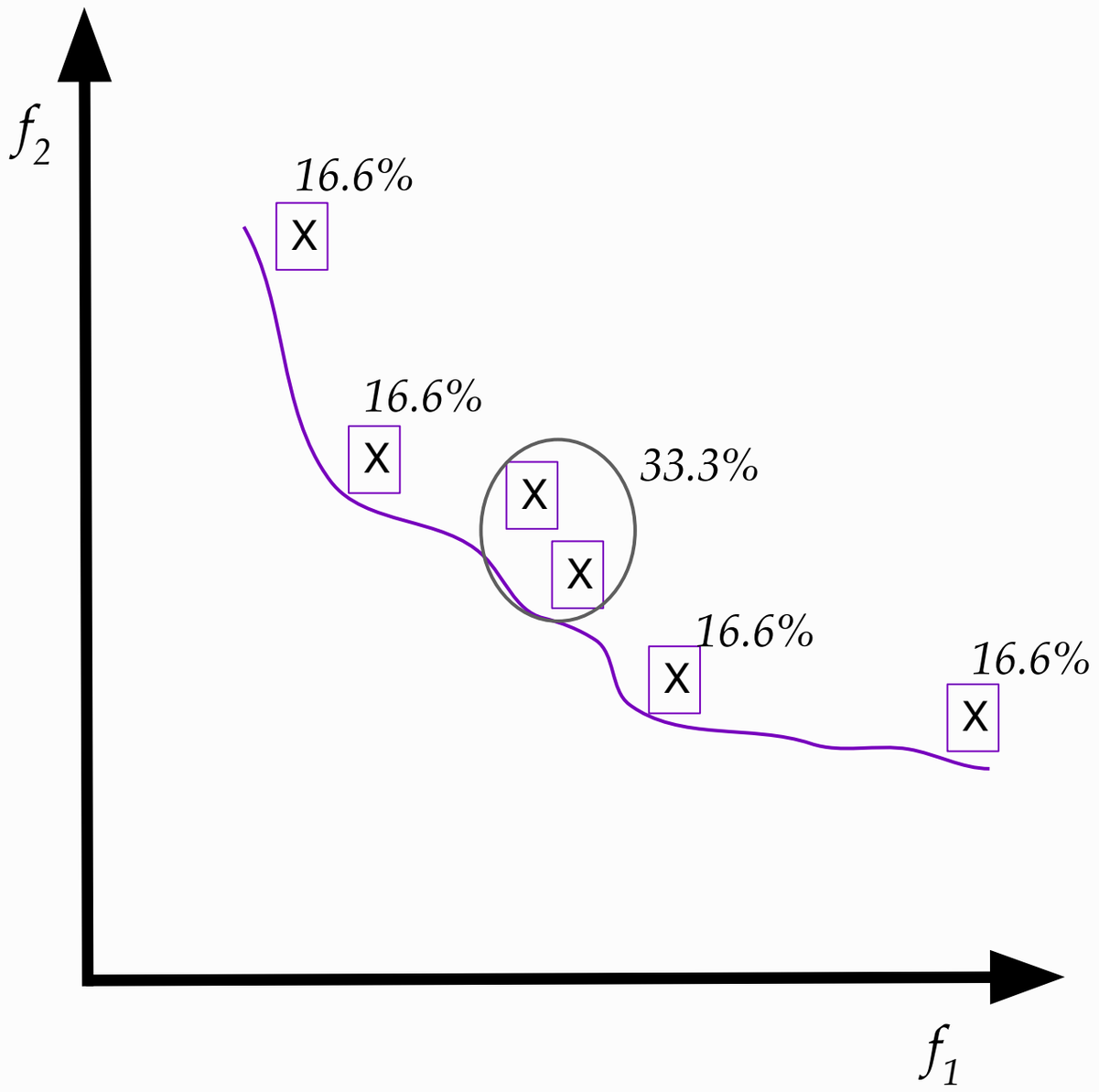
Soluzione di NSGA

Ritorniamo all'esempio precedente, ed immaginiamo di selezionare il Fronte 3. Ciascun individuo ha circa il 16.6% di probabilità di essere selezionato.

Selezionare individui con trade-off troppo simile **riduce la diversità** che ci piace.

Inoltre, aumenta il rischio di convergere prematuramente.

Ci piacerebbe un meccanismo per favorire le soluzioni più "distanti" dalle altre.



Algoritmi Genetici

GA per MOO - Selezione

Soluzione di NSGA

Soluzione intelligente. Si favoriscono gli individui più “isolati dalla folla”, tramite una *misura di distanza* chiamata **crowding distance**. Si opera nel seguente modo:

Per ogni fronte:

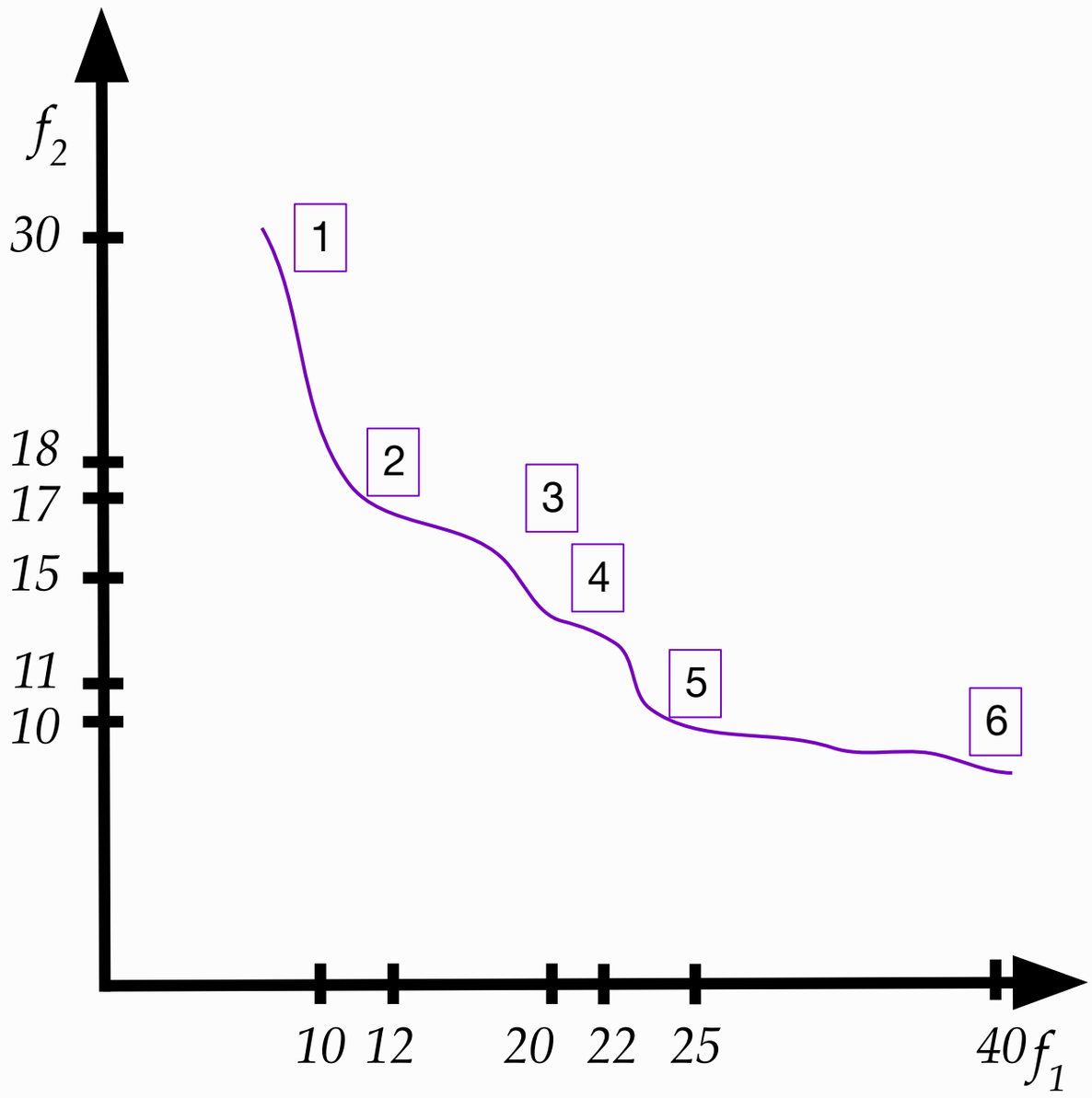
1. Si valuta ogni individuo rispetto a ciascuna funzione di valutazione;
2. Per ogni funzione di valutazione f_i:
   a. si ordinano gli individui rispetto al valore f_i;
   b. si assegna al primo e all’ultimo individuo una **distanza infinita**;
   c. si itera sugli altri individui, e si assegna a ciascuno una distanza pari alla **differenza delle valutazioni f_i delle due soluzioni adiacenti** (nell’ordinamento), **normalizzata sulle f_i estreme (del primo e dell’ultimo individuo)**.
3. Per ogni individuo, si sommano tutte le distanze ricevute, ottenendo così la **crowding distance** (se la somma coinvolge almeno un valore infinito, allora la crowding distance sarà infinita).

Come si interpreta questa misura? **Tanto più è piccola, tanto più l’individuo è vicino ad altri**, quindi da preferire di meno perché darebbe luogo ad una minore diversità.

GA per MOO - Selezione

Soluzione di NSGA

Vediamo un esempio geometricamente impreciso. Selezioniamo di nuovo il Fronte 3.



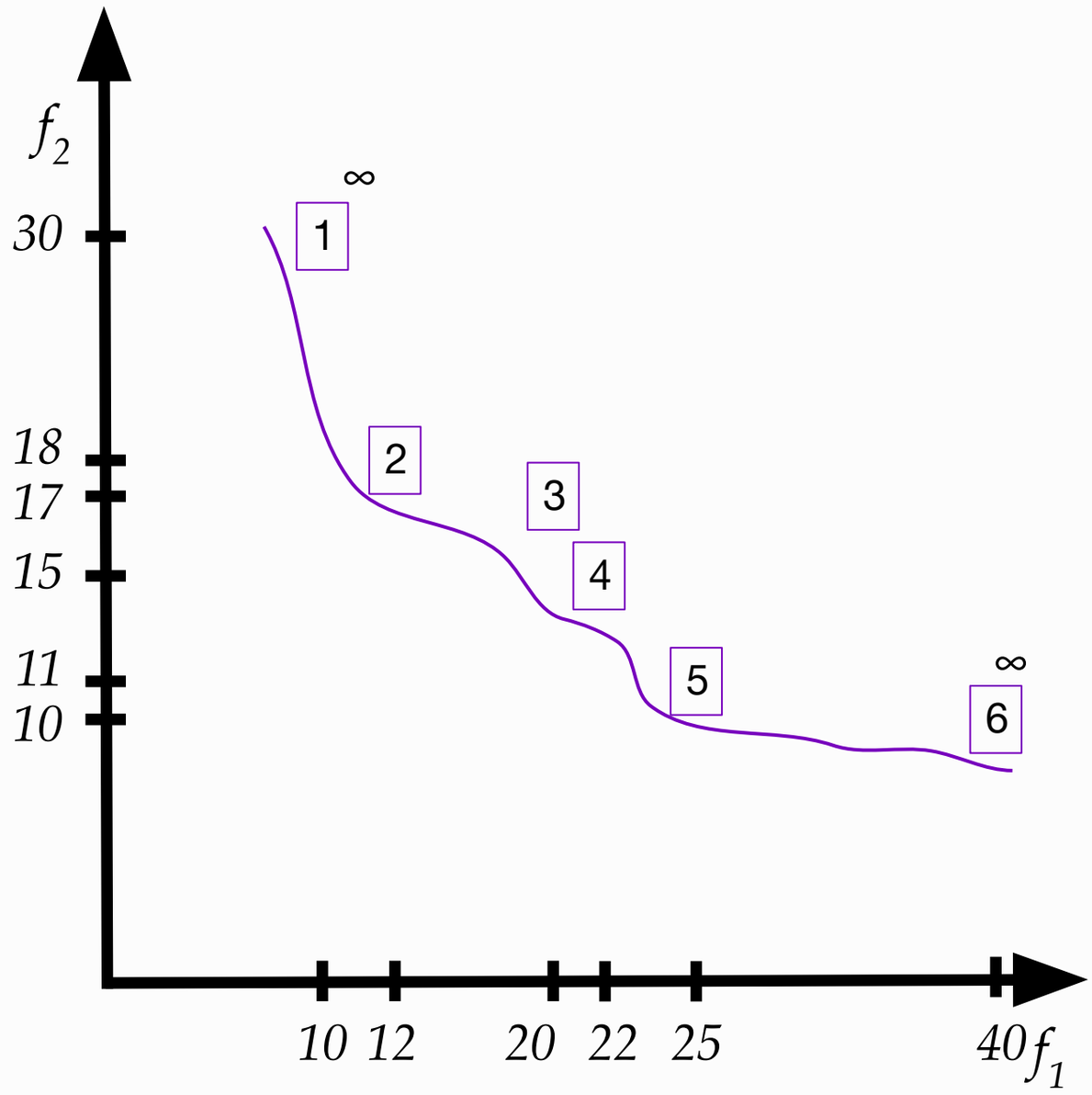
Algoritmi Genetici

GA per MOO - Selezione

Soluzione di NSGA

Vediamo un esempio geometricamente impreciso. Selezioniamo di nuovo il Fronte 3.

Gli individui 1 e 6 ricevono crowding distance infinita (infinito per almeno una funzione).



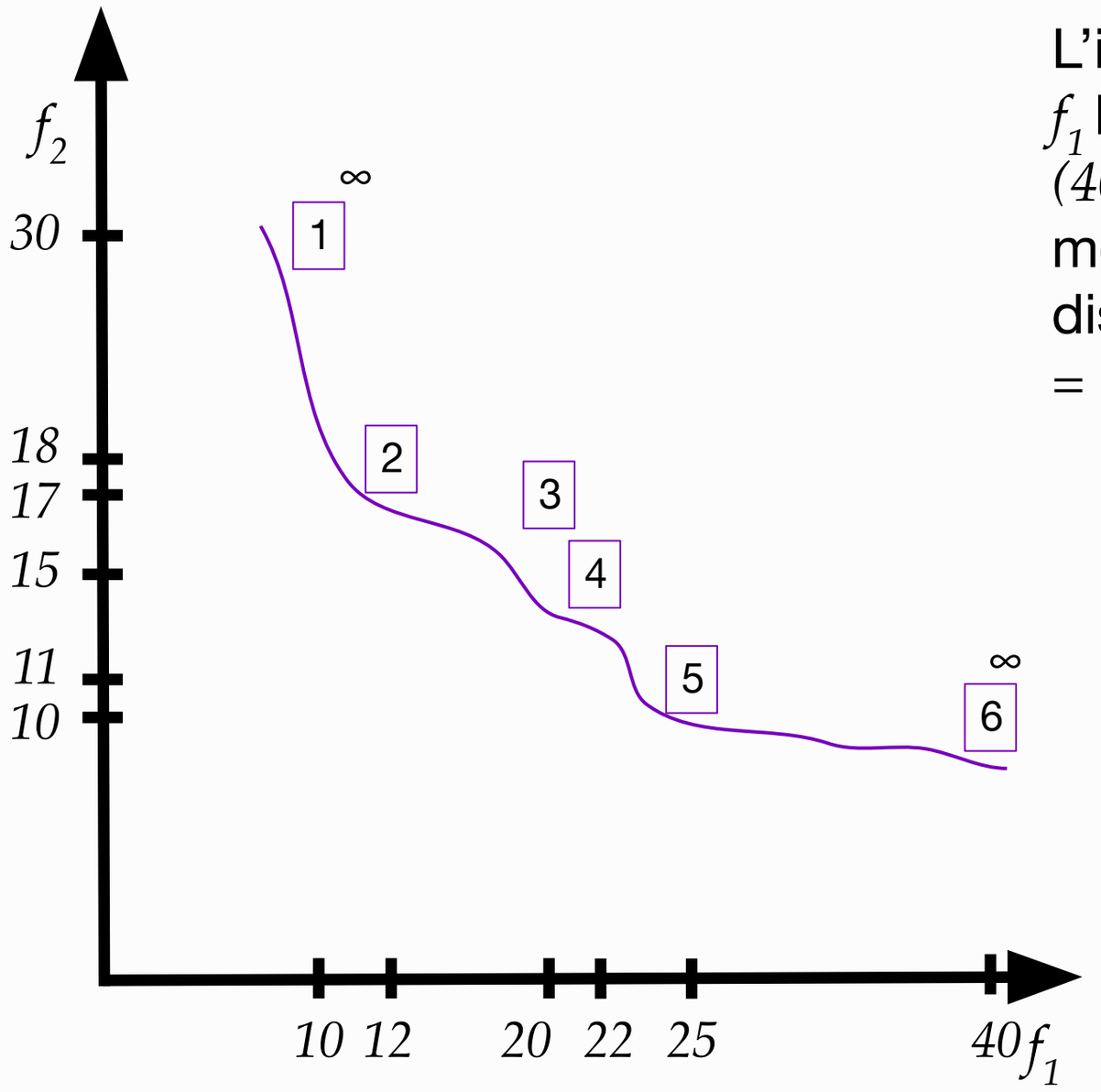
Algoritmi Genetici

GA per MOO - Selezione

Soluzione di NSGA

Vediamo un esempio geometricamente impreciso. Selezioniamo di nuovo il Fronte 3.

Gli individui 1 e 6 ricevono crowding distance infinita (infinito per almeno una funzione).



L'individuo 2 rispetto ad f_1 ha distanza *(20-10) / (40-10) = 10/30 = 0.3*, mentre per f_2 ha distanza *(30-17) / (30-10) = 13/20 = 0.65*.

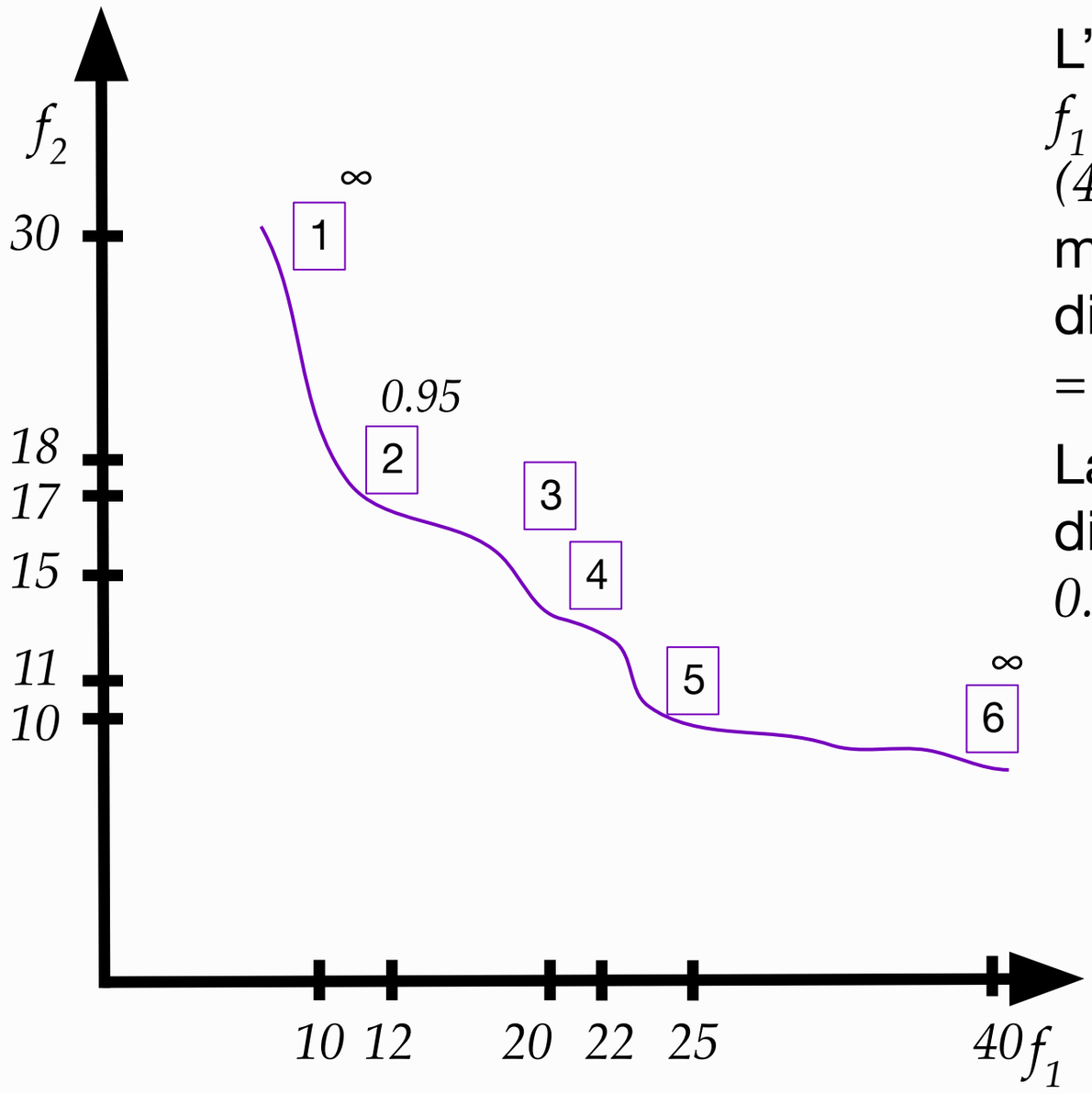
Algoritmi Genetici

GA per MOO - Selezione

Soluzione di NSGA

Vediamo un esempio geometricamente impreciso. Selezioniamo di nuovo il Fronte 3.

Gli individui 1 e 6 ricevono crowding distance infinita (infinito per almeno una funzione).



L'individuo 2 rispetto ad f_1 ha distanza $(20\text{-}10) / (40\text{-}10) = 10/30 = 0.3$, mentre per f_2 ha distanza $(30\text{-}17) / (30\text{-}10) = 13/20 = 0.65$.

La sua crowding distance risulta pari a $0.3 + 0.65 = 0.95$.

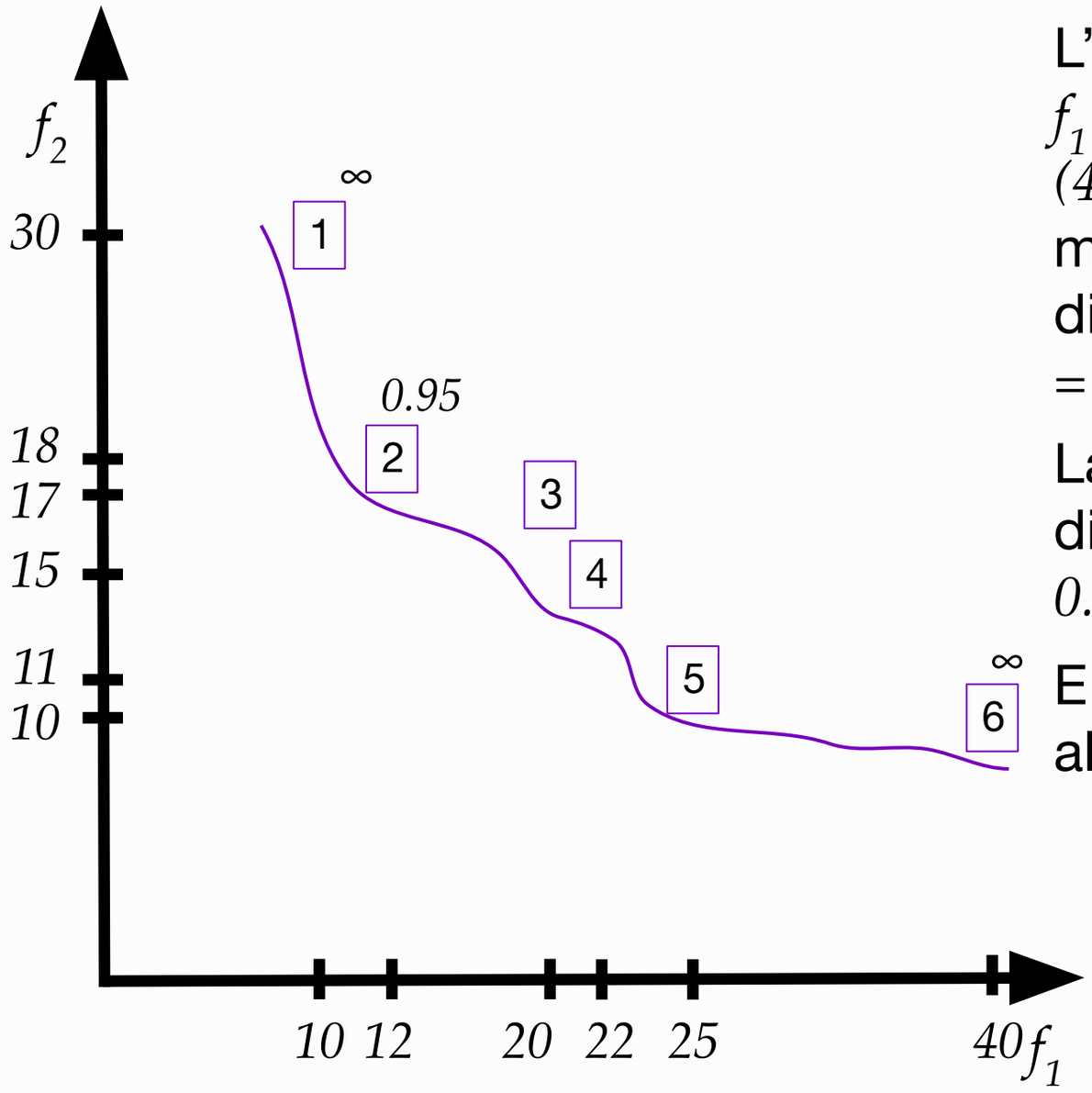
Algoritmi Genetici

GA per MOO - Selezione

Soluzione di NSGA

Vediamo un esempio geometricamente impreciso. Selezioniamo di nuovo il Fronte 3.

Gli individui 1 e 6 ricevono crowding distance infinita (infinito per almeno una funzione).



L'individuo 2 rispetto ad f_1 ha distanza $(20\text{-}10) / (40\text{-}10) = 10/30 = 0.3$, mentre per f_2 ha distanza $(30\text{-}17) / (30\text{-}10) = 13/20 = 0.65$.

La sua crowding distance risulta pari a $0.3 + 0.65 = 0.95$.

E così via per tutti gli altri individui...

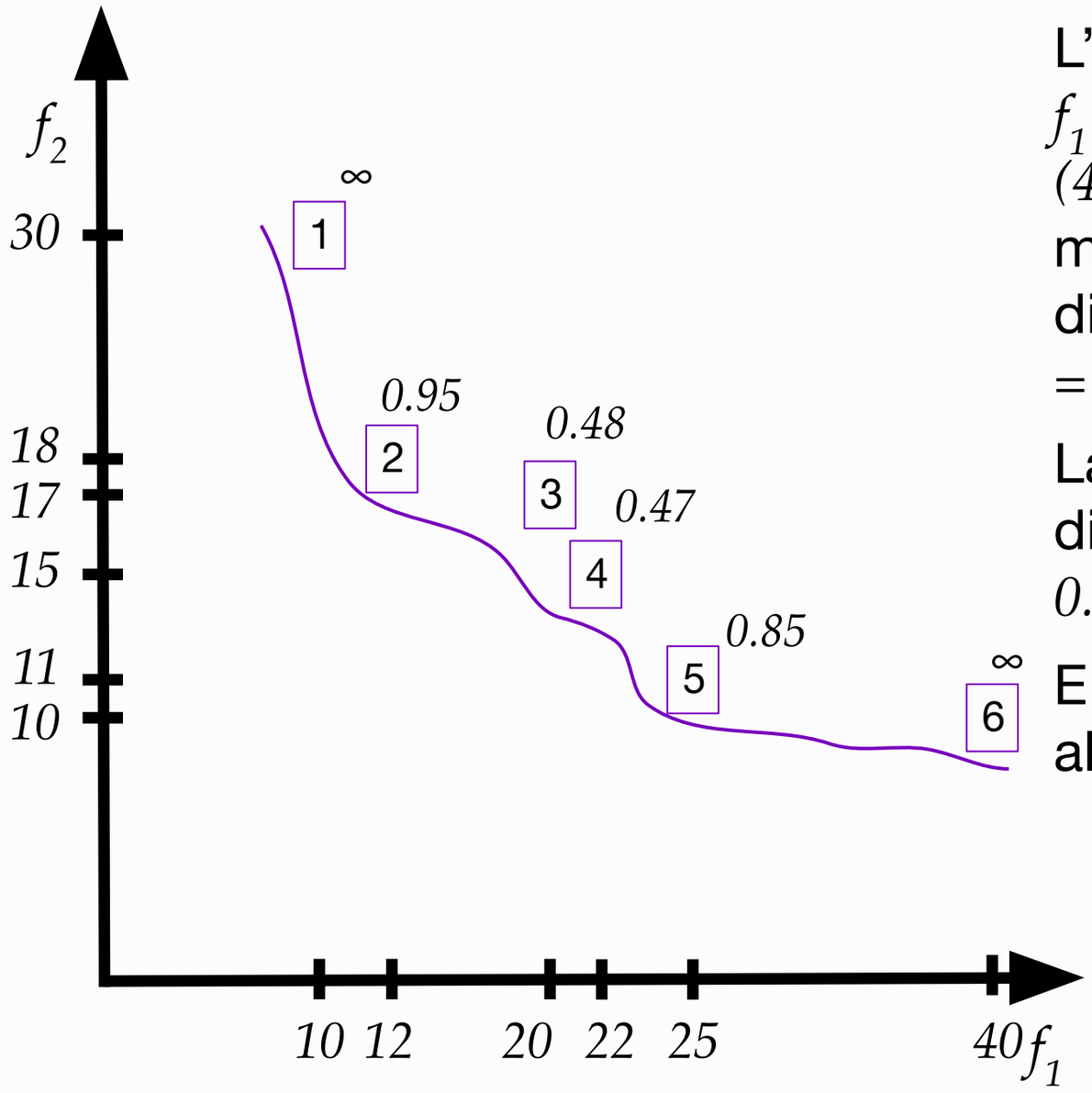
Algoritmi Genetici

GA per MOO - Selezione

Soluzione di NSGA

Vediamo un esempio geometricamente impreciso. Selezioniamo di nuovo il Fronte 3.

Gli individui 1 e 6 ricevono crowding distance infinita (infinito per almeno una funzione).



L'individuo 2 rispetto ad f_1 ha distanza $(20\text{-}10) / (40\text{-}10) = 10/30 = 0.3$, mentre per f_2 ha distanza $(30\text{-}17) / (30\text{-}10) = 13/20 = 0.65$.

La sua crowding distance risulta pari a $0.3 + 0.65 = 0.95$.

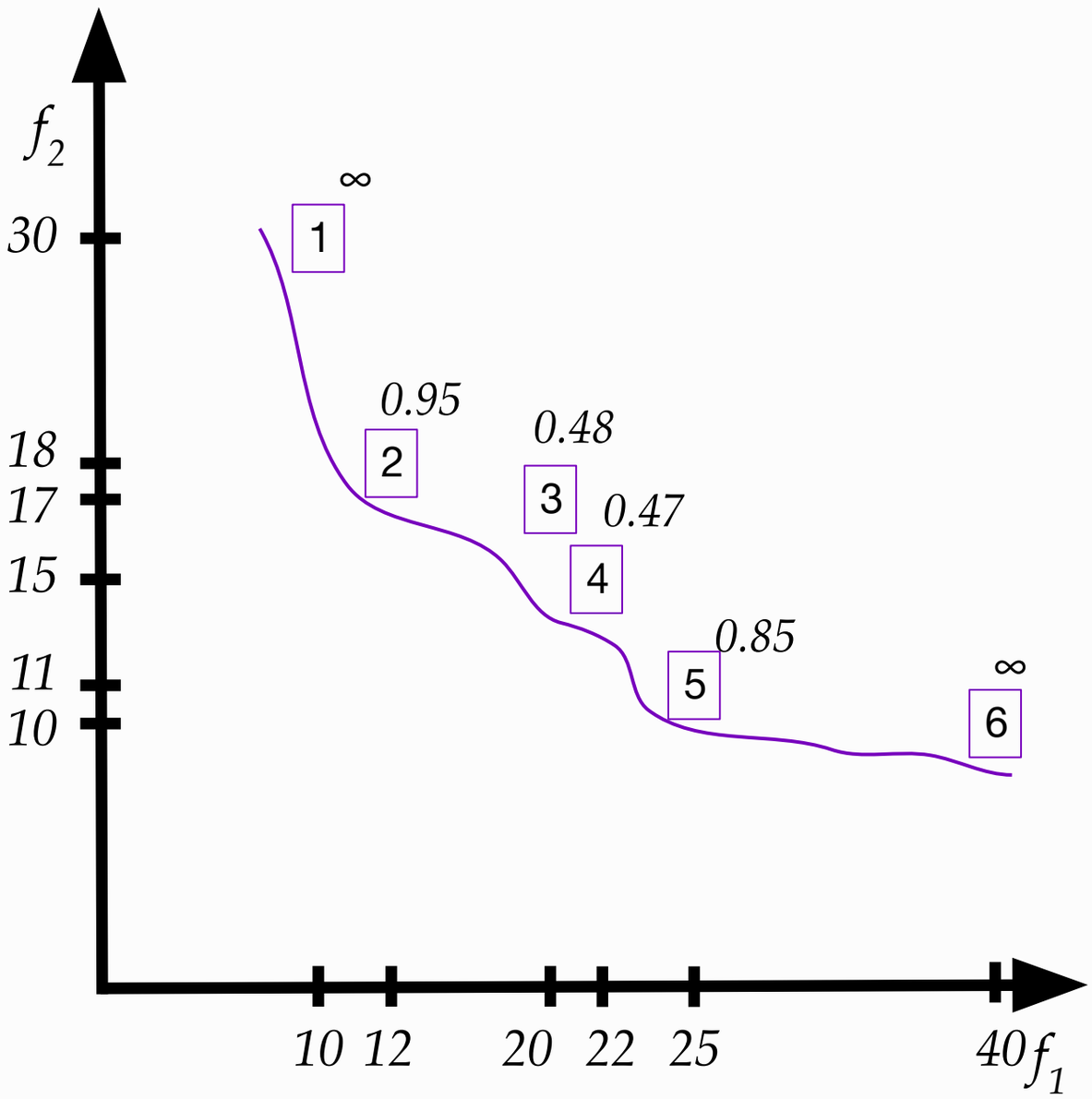
E così via per tutti gli altri individui...

Algoritmi Genetici

GA per MOO - Selezione

Soluzione di NSGA

Vediamo un esempio geometricamente impreciso. Selezioniamo di nuovo il Fronte 3.



A questo punto abbiamo ottenuto un **nuovo criterio di ordinamento** che guida l'algoritmo di selezione nel prendere la decisione su chi deve favorire.

Nello specifico, dati due individui qualsiasi x e y, la selezione viene vinta da x se ha un **livello di dominanza inferiore** OPPURE ha una **crowding distance superiore**.

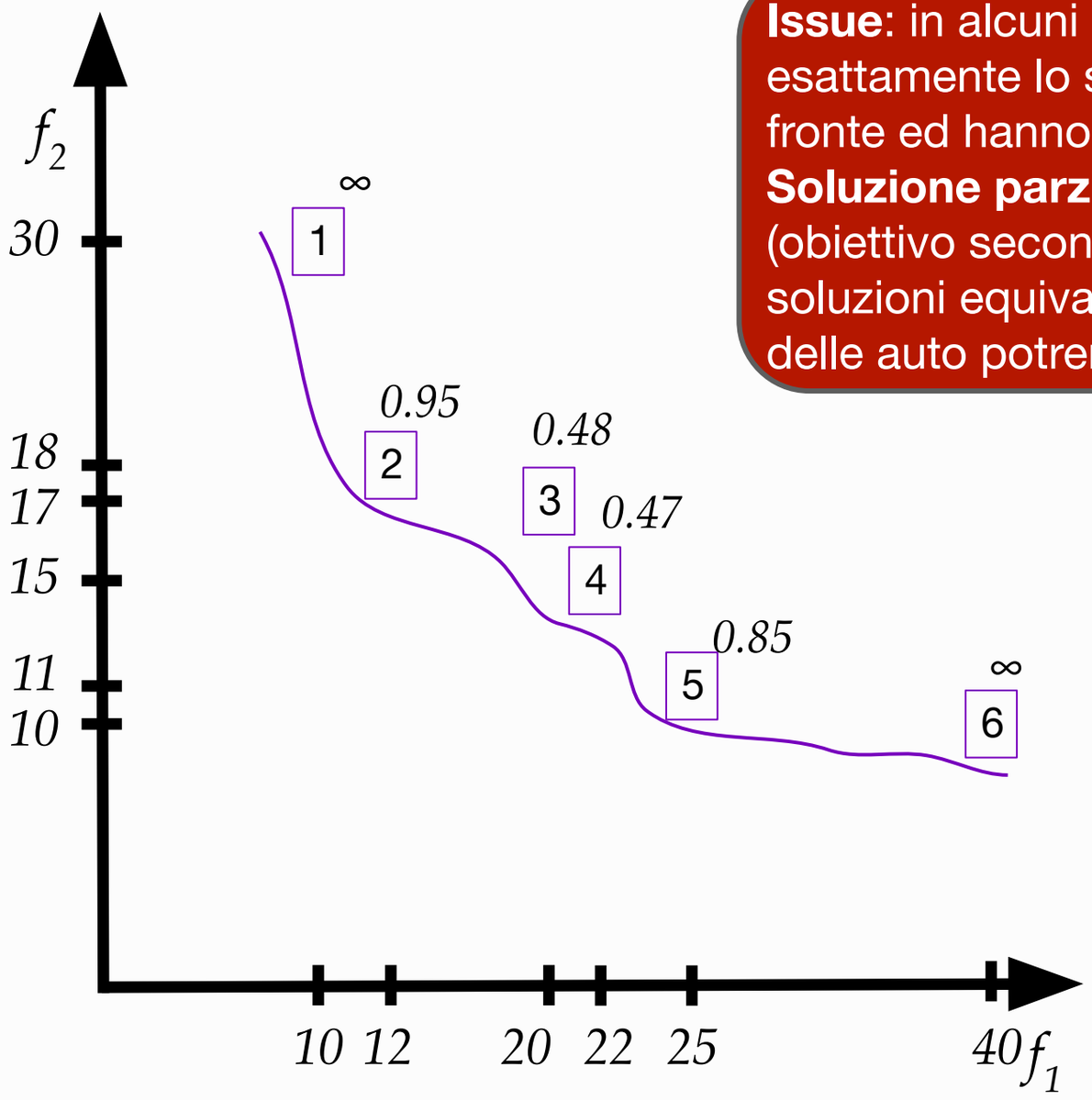
Con queste due regole non è più necessario adottare una funzione di fitness. E' comunque possibile, ma ci impone di definire un'aggregazione del livello di dominanza con la crowding distance. Per questo motivo, si evitano selezioni fitness-proportionate in favore di altre quali la Tournament Selection.

Algoritmi Genetici

GA per MOO - Selezione

Soluzione di NSGA

Vediamo un esempio geometricamente impreciso. Selezioniamo di nuovo il Fronte 3.



Issue: in alcuni problemi non è difficile avere individui con esattamente lo stesso vettore di valutazione (finiscono nello stesso fronte ed hanno anche la stessa crowding distance).
Soluzione parziale: si introduce un **criterio di preferenza** (obiettivo secondario problem-specific) per "disambiguare" le soluzioni equivalenti, ad esempio nel trade-off tra comfort e costo delle auto potremmo considerare l'anno di costruzione dell'auto.

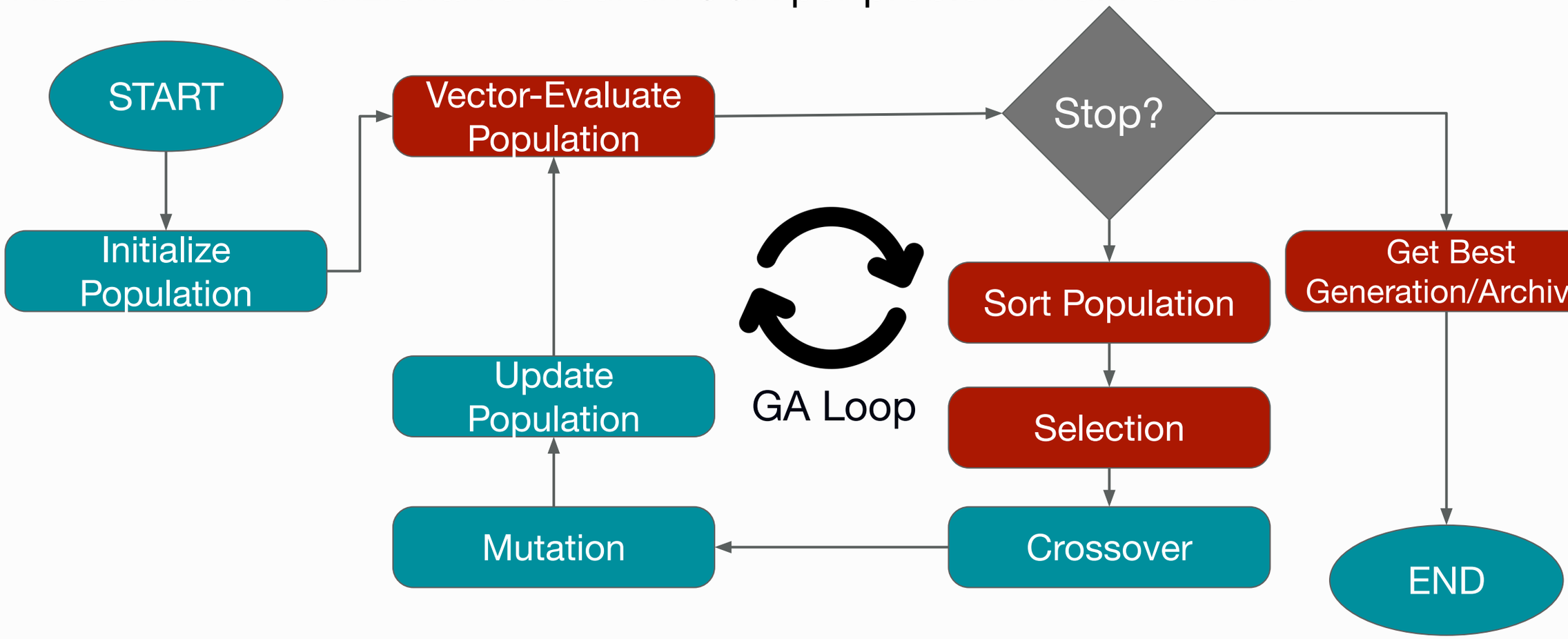
x e y, la selezione vien[illegible] x se ha un **livello di dominanza inferiore** OPPURE ha una **crowding distance superiore.**

Con queste due regole non è più necessario adottare una funzione di fitness. E' comunque possibile, ma ci impone di definire un'aggregazione del livello di dominanza con la crowding distance. Per questo motivo, si evitano selezioni fitness-proportionate in favore di altre quali la Tournament Selection.

Algoritmi Genetici

GA per MOO - Recap

Riassumiamo il funzionamento di un SGA per problemi multi-obiettivo:



Algoritmi Genetici

Conclusioni - Varianti dei GA

Ci sarebbe molto altro da dire sugli algoritmi genetici. Concludiamo illustrando le principali **varianti** che portano gli algoritmi genetici ad essere potenziati o a risolvere nuove classi di problemi.

Steady-State GA

Anche noto come *Genitor-style GA*. Invece di creare nuove generazioni modifica continuamente la stessa con miglioramenti graduali. Infatti:

1. vengono scelti **i migliori due individui** (selezione basata su ranghi) come genitori;
2. con il crossover si generano un solo figlio, soggetto poi a mutazione;
3. il figlio è valutato e andrà, eventualmente, a **rimpiazzare il peggior individuo**.

Evolve "lentamente" ma garantisce un **miglioramento monotòno**.

Parallel GA

I GA si prestano per alcune efficienti implementazioni parallele. Vediamo le 3 principali:

Global Parallelism | **Island Model GA** | **Cellular GA**

Con una popolazione di taglia N (pari), $N/2$ processori, ed una selezione di tipo Tournament Selection, applichiamo i seguenti passi:

1. si distribuiscono le $N/2$ coppie di individui a ciascun processore;
2. ciascun processore valuta i due individui ed applica selezione;
3. si ripetono i passi 1 e 2 per una seconda volta;
4. ciascun processore applica il crossover e la mutazione dei due figli;
5. si raccolgono i figli da ciascun processore, formando la nuova generazione.

Algoritmi Genetici

Conclusioni - Varianti dei GA

Ci sarebbe molto altro da dire sugli algoritmi genetici. Concludiamo illustrando le principali **varianti** che portano gli algoritmi genetici ad essere potenziati o a risolvere nuove classi di problemi.

Steady-State GA

Anche noto come *Genitor-style GA*. Invece di creare nuove generazioni modifica continuamente la stessa con miglioramenti graduali. Infatti:

1. vengono scelti **i migliori due individui** (selezione basata su ranghi) come genitori;
2. con il crossover si generano un solo figlio, soggetto poi a mutazione;
3. il figlio è valutato e andrà, eventualmente, a **rimpiazzare il peggior individuo**.

Evolve "lentamente" ma garantisce un **miglioramento monotòno**.

Parallel GA

I GA si prestano per alcune efficienti implementazioni parallele. Vediamo le 3 principali:

Global Parallelism

Island Model GA

Cellular GA

Con una popolazione di taglia N (molto grande) ed M processori possiamo assegnare a ciascun processore una **sotto-popolazione di N/M individui**:

Ciascun processore esegue un qualsiasi GA (SGA, Steady-State GA, ecc.) in maniera indipendente dalle altre (rappresenta quindi "un'isola"). Per evitare **derive genetiche** (quando delle popolazioni piccole tendono ad evolvere verso una direzione specifica, non necessariamente ottimale), ogni K generazioni (ad esempio, ogni 5) le sotto-popolazioni si scambiano alcuni individui: si applica la cosiddetta **migrazione**.

Algoritmi Genetici

Conclusioni - Varianti dei GA

Ci sarebbe molto altro da dire sugli algoritmi genetici. Concludiamo illustrando le principali **varianti** che portano gli algoritmi genetici ad essere potenziati o a risolvere nuove classi di problemi.

Steady-State GA

Anche noto come *Genitor-style GA*. Invece di creare nuove generazioni modifica continuamente la stessa con miglioramenti graduali. Infatti:

1. vengono scelti **i migliori due individui** (selezione basata su ranghi) come genitori;
2. con il crossover si generano un solo figlio, soggetto poi a mutazione;
3. il figlio è valutato e andrà, eventualmente, a **rimpiazzare il peggior individuo**.

Evolve "lentamente" ma garantisce un **miglioramento monotòno**.

Parallel GA

I GA si prestano per alcune efficienti implementazioni parallele. Vediamo le 3 principali:

Global Parallelism | **Island Model GA** | **Cellular GA**

Con una popolazione di taglia N (molto grande), si crea una **matrice** $\sqrt{N}x\sqrt{N}$ di individui. Ogni elemento (**cella**) viene assegnato ad un singolo processore.

Nella variante standard, ogni cella può comunicare **soltanto con le celle adiacenti** (nord, sud, est, ed ovest, in modulo). Ad ogni iterazione, ogni cella valuta il proprio individuo, si accoppia con l'individuo vicino più forte (o usando una accoppiamento probabilistico), si genera un singolo figlio, lo si muta, e si rimpiazza il genitore se risulta migliore. Gruppi molto molto distanti di celle formano delle sotto-popolazioni (*isolamento per distanza*).

Conclusioni - Varianti dei GA

Ci sarebbe molto altro da dire sugli algoritmi genetici. Concludiamo illustrando le principali **varianti** che portano gli algoritmi genetici ad essere potenziati o a risolvere nuove classi di problemi.

Algoritmi Memetici

Introdurre della ricerca locale all'interno del GA loop (*ibridizzazione*) allo scopo di ridurre il rischio di convergenza prematura. In particolare, dopo aver applicato la mutazione alla nuova generazione di individui si seleziona un **subset casuale degli individui che sarà sottoposto ad una fase di ricerca locale** (ad esempio, Hill Climbing), modificandosi in meglio. Questa fase di *raffinamento locale* rappresenta un'**evoluzione culturale**.

Algoritmi di questo tipo sono anche noti come *Algoritmi Evolutivi Lamarckiani*, che ricalcano la teoria evoluzionistica di Lamarck, che afferma che le **caratteristiche più o meno utilizzate** di un individuo si trasmettono ai figli, cosa non vera secondo Darwin.

Per ricalcare ulteriormente l'evoluzione lamarckiana, gli **algoritmi memetici co-evolutivi** considerano i *meme* (secondo Dawkins, equivalenti dei geni ma in un contesto di evoluzione culturale) insieme alla componente genetica di un individuo. La componente memetica codificano il miglioramento ottenuto dall'ottimizzazione locale, e partecipa anche essa all'evoluzione. Infatti, contribuisce alla funzione di valutazione ed è viene *trasmessa* alla prole, in maniera analoga al crossover genetico.

UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Laurea triennale in Informatica
Anno accademico 2021/2022

Fondamenti di Intelligenza Artificiale

Lezione 8 - Algoritmi di ricerca locale (III)