Direct Multisearch

Optimization of Complex Systems – March 24th 2026

Andrea Brilli

Sapienza University of Rome

Recap: where we are

In the last two classes we built the theory of multiobjective optimization:

- Pareto dominance ($\preceq$) and strict dominance ($\prec$) in $\mathbb{R}^p$
- The Pareto front Ω_P and weak Pareto front Ω_D
- First-order optimality via the **stationarity measure**:

$$\theta(x) = -\min_{\|d\|\leq 1} \max_{i=1,\ldots,p} \nabla f_i(x)^\top d \geq 0, \quad \theta(x) = 0 \iff x \text{ is Pareto stationary}$$

- **Scalarization approaches**: weighted sum, ε-constraint

Recap: where we are

In the last two classes we built the theory of multiobjective optimization:

- Pareto dominance ($\preceq$) and strict dominance ($\prec$) in $\mathbb{R}^p$
- The Pareto front Ω_P and weak Pareto front Ω_D
- First-order optimality via the **stationarity measure**:

$$\theta(x) = -\min_{\|d\|\leq 1} \max_{i=1,\ldots,p} \nabla f_i(x)^\top d \geq 0, \quad \theta(x) = 0 \iff x \text{ is Pareto stationary}$$

- **Scalarization approaches**: weighted sum, ε-constraint

Common feature of all scalarization methods: each converts the multiobjective problem into a *single-objective* one, parameterized by weights or thresholds.

Question: is this reduction always desirable? What are we *implicitly* assuming when we scalarize?

The price of scalarization

Each scalarization method converts

$$\min_{x \in \mathbb{R}^n} \; (f_1(x), \ldots, f_p(x))$$

into a scalar problem parameterized by user-supplied information (λ, ε_i, ...).

What can go wrong?

- Both methods require **a priori preference information** that may be unavailable
- To approximate the *entire* front, one must solve **many scalar problems** sequentially — with no coordination between solves
- The resulting points are **loosely coupled**: nothing guarantees a good coverage of Ω_P

The price of scalarization

Each scalarization method converts

$$\min_{x \in \mathbb{R}^n} (f_1(x), \ldots, f_p(x))$$

into a scalar problem parameterized by user-supplied information (λ, ε_i, ...).

What can go wrong?

- Both methods require **a priori preference information** that may be unavailable
- To approximate the *entire* front, one must solve **many scalar problems** sequentially — with no coordination between solves
- The resulting points are **loosely coupled**: nothing guarantees a good coverage of Ω_P

The price of scalarization

Question: can we design an algorithm that *directly* approximates the Pareto front, without ever collapsing the problem to a scalar one?

Yes. This is the contribution of:

"Direct Multisearch for Multiobjective Optimization"
Custódio, Madeira, Vaz, Vicente (2011)

Before looking at the algorithm, let us ask a more fundamental question: *why did this idea first appear in a* **derivative-free** *context?*

A natural extension of steepest descent

In single-objective optimization, steepest descent generates a sequence $\{x_k\}$ by moving in the direction $d = -\nabla f(x_k)/\|\nabla f(x_k)\|$.

Question: how would you generalize this to the multiobjective case? We have p gradients, $\nabla f_1(x), \ldots, \nabla f_p(x)$. What is the "right" direction to move?

A natural extension of steepest descent

Recall: the stationarity measure $\theta(x)$ provides exactly the answer.

If $\theta(x_k) > 0$, the direction

$$d^*(x_k) \;=\; \arg \min_{\|d\| \leq 1} \; \max_{i=1,\ldots,p} \; \nabla f_i(x_k)^\top d$$

satisfies $-\nabla f_i(x_k)^\top d^*(x_k) \geq \theta(x_k) > 0$ for **all** $i = 1, \ldots, p$ simultaneously.

A natural extension of steepest descent

Recall: the stationarity measure $\theta(x)$ provides exactly the answer.

If $\theta(x_k) > 0$, the direction

$$d^*(x_k) \;=\; \arg \min_{\|d\| \leq 1} \; \max_{i=1,\ldots,p} \; \nabla f_i(x_k)^\top d$$

satisfies $-\nabla f_i(x_k)^\top d^*(x_k) \geq \theta(x_k) > 0$ for **all** $i = 1, \ldots, p$ simultaneously.

Consequence: for small enough $\alpha > 0$,

$$f_i(x_k + \alpha d^*(x_k)) \;<\; f_i(x_k), \quad \forall\, i = 1, \ldots, p,$$

i.e., $f(x_k + \alpha d^*(x_k)) \prec f(x_k)$: the new point **strictly dominates** the current one.

This is the multiobjective analogue of gradient descent: an *implicit scalarization* through θ.

Success and failure in gradient-based MO descent

A gradient-based multiobjective steepest descent step:

1. Compute $\theta(x_k)$ and $d^*(x_k)$
2. Find $\alpha_k > 0$ with sufficient decrease: $f(x_k + \alpha_k d^*) \prec f(x_k)$
3. Set $x_{k+1} = x_k + \alpha_k d^*$; if no such α_k exists, reduce step-size
4. Stop when $\theta(x_k) = 0$

Success and failure in gradient-based MO descent

A gradient-based multiobjective steepest descent step:

1. Compute $\theta(x_k)$ and $d^*(x_k)$
2. Find $\alpha_k > 0$ with sufficient decrease: $f(x_k + \alpha_k d^*) \prec f(x_k)$
3. Set $x_{k+1} = x_k + \alpha_k d^*$; if no such α_k exists, reduce step-size
4. Stop when $\theta(x_k) = 0$

The notion of success is clear: a step is successful if and only if $f(x_{k+1}) \prec f(x_k)$, i.e., the new point **strictly dominates** the old one.

Success and failure in gradient-based MO descent

A gradient-based multiobjective steepest descent step:

1. Compute $\theta(x_k)$ and $d^*(x_k)$
2. Find $\alpha_k > 0$ with sufficient decrease: $f(x_k + \alpha_k d^*) \prec f(x_k)$
3. Set $x_{k+1} = x_k + \alpha_k d^*$; if no such α_k exists, reduce step-size
4. Stop when $\theta(x_k) = 0$

The notion of success is clear: a step is successful if and only if $f(x_{k+1}) \prec f(x_k)$, i.e., the new point **strictly dominates** the old one.

Question: suppose we stumble upon a point $\tilde{x}$ such that $f(\tilde{x})$ and $f(x_k)$ are **mutually nondominated**:

$$f(\tilde{x}) \not\preceq f(x_k) \quad \text{and} \quad f(x_k) \not\preceq f(\tilde{x}).$$

In a gradient-based method with $\theta(x_k) \neq 0$, is this a success or a failure?

A nondominated point is a failure — when you have gradients

If $\theta(x_k) \neq 0$, then by definition there exists a direction $d^*(x_k)$ along which *all* objectives decrease simultaneously.

This means: there exist points arbitrarily close to x_k that *strictly dominate* $f(x_k)$.

A point $\tilde{x}$ that is merely *nondominated* (but does not dominate x_k) is therefore a **failure**: the gradient is telling us we could have found better.

The gradient is the oracle. It says: "$\theta(x_k) < 0$ means a strictly dominating step is possible — anything less is a missed opportunity."

A nondominated point is a failure — when you have gradients

If $\theta(x_k) \neq 0$, then by definition there exists a direction $d^*(x_k)$ along which *all* objectives decrease simultaneously.

This means: there exist points arbitrarily close to x_k that *strictly dominate* $f(x_k)$.

A point $\tilde{x}$ that is merely *nondominated* (but does not dominate x_k) is therefore a **failure**: the gradient is telling us we could have found better.

The gradient is the oracle. It says: "$\theta(x_k) < 0$ means a strictly dominating step is possible — anything less is a missed opportunity."

A nondominated point is a failure — when you have gradients

Trial outcome	Gradient-based verdict	Why
$f(\tilde{x}) \prec f(x_k)$	Success	$\tilde{x}$ dominates x_k
$f(\tilde{x}) \not\prec f(x_k)$	Failure	$\theta(x_k) < 0 \Rightarrow$ better exists

In particular, a **nondominated-but-not-dominating** point is *discarded*.

This logic comes from single-objective thinking: maintain a *single-point sequence*, strictly improved at every step. The multi-objective gradient method simply upgrades "improvement" to "domination".

What changes without gradients?

Now suppose we are in a **derivative-free** (black-box) setting: $f(x)$ is available, but $\nabla f_i(x)$ is not.

Question: without computing $\theta(x_k)$, can we still declare a mutually nondominated point $\tilde{x}$ a "failure"?

We cannot — and here is the key reason:

- We do not know whether $\theta(x_k) < 0$ or $\theta(x_k) = 0$
- If $\theta(x_k) = 0$, then x_k **is** Pareto stationary — perhaps even Pareto optimal
- A nondominated point $\tilde{x}$ found near a Pareto stationary x_k could itself be **another Pareto optimal point**
- We have no oracle that certifies we "could have done better"

What changes without gradients?

Now suppose we are in a **derivative-free** (black-box) setting: $f(x)$ is available, but $\nabla f_i(x)$ is not.

Question: without computing $\theta(x_k)$, can we still declare a mutually nondominated point $\tilde{x}$ a "failure"?

We cannot — and here is the key reason:

- We do not know whether $\theta(x_k) < 0$ or $\theta(x_k) = 0$
- If $\theta(x_k) = 0$, then x_k **is** Pareto stationary — perhaps even Pareto optimal
- A nondominated point $\tilde{x}$ found near a Pareto stationary x_k could itself be **another Pareto optimal point**
- We have no oracle that certifies we "could have done better"

What changes without gradients?

Now suppose we are in a **derivative-free** (black-box) setting: $f(x)$ is available, but $\nabla f_i(x)$ is not.

Question: without computing $\theta(x_k)$, can we still declare a mutually nondominated point $\tilde{x}$ a "failure"?

We cannot — and here is the key reason:

- We do not know whether $\theta(x_k) < 0$ or $\theta(x_k) = 0$
- If $\theta(x_k) = 0$, then x_k **is** Pareto stationary — perhaps even Pareto optimal
- A nondominated point $\tilde{x}$ found near a Pareto stationary x_k could itself be **another Pareto optimal point**
- We have no oracle that certifies we "could have done better"

Discarding $\tilde{x}$ would be **unjustified**: in the absence of gradient information, *nondominated* is as good as we can ask for.

The key insight: keep both points

Question: if we cannot decide whether $\tilde{x}$ is "better" than x_k, what should we do?

Keep both.

Since $f(\tilde{x})$ and $f(x_k)$ are mutually nondominated, neither dominates the other, and both could be near the Pareto front. We have no rational ground to discard either.

This simple observation forces a completely different algorithmic architecture: instead of maintaining a **single iterate**, maintain a **list of nondominated points** — an *archive* $\mathcal{L}_k$.

The key insight: keep both points

Question: if we cannot decide whether $\tilde{x}$ is "better" than x_k, what should we do?

Keep both.

Since $f(\tilde{x})$ and $f(x_k)$ are mutually nondominated, neither dominates the other, and both could be near the Pareto front. We have no rational ground to discard either.

This simple observation forces a completely different algorithmic architecture: instead of maintaining a **single iterate**, maintain a **list of nondominated points** — an *archive* $\mathcal{L}_k$.

The key insight: keep both points

The archive $\mathcal{L}_k$: at each iteration k, a set of *mutually nondominated* iterates found so far.

Each iteration:

1. Choose a **polling center** $x^{(j)} \in \mathcal{L}_k$
2. Evaluate f at trial points around $x^{(j)}$ (on a mesh)
3. **Accept** any trial point $\tilde{x}$ not dominated by any $z \in f(\mathcal{L}_k)$
4. Update $\mathcal{L}_{k+1}$: add accepted points, remove points now dominated

The key insight: keep both points

The archive $\mathcal{L}_k$: at each iteration k, a set of *mutually nondominated* iterates found so far.

Each iteration:

1. Choose a **polling center** $x^{(j)} \in \mathcal{L}_k$
2. Evaluate f at trial points around $x^{(j)}$ (on a mesh)
3. **Accept** any trial point $\tilde{x}$ not dominated by any $z \in f(\mathcal{L}_k)$
4. Update $\mathcal{L}_{k+1}$: add accepted points, remove points now dominated

Output: not a single point, but a finite **approximation of the Pareto front**.

The algorithm does *not* need to know whether it has "converged" to a single solution — convergence means the *archive* approximates Ω_P.

Why did this idea arise in derivative-free first?

This is a question worth pausing on.

Question: gradient-based methods predate derivative-free ones. Why did the archive idea appear *first* in the derivative-free literature, not the gradient-based one?

Why did this idea arise in derivative-free first?

This is a question worth pausing on.

The gradient-based bias:

- Gradient-based MO methods are natural extensions of steepest descent: compute $\theta(x_k)$, move in $d^*(x_k)$, maintain a *single-point sequence*
- This paradigm is *so natural* that the question "what if we keep multiple points?" is never forced by the mathematics
- A nondominated non-dominating point is *logically* a failure: $\theta(x_k) < 0$ proves we could have done better

Why did this idea arise in derivative-free first?

This is a question worth pausing on.

The gradient-based bias:

- Gradient-based MO methods are natural extensions of steepest descent: compute $\theta(x_k)$, move in $d^*(x_k)$, maintain a *single-point sequence*
- This paradigm is *so natural* that the question "what if we keep multiple points?" is never forced by the mathematics
- A nondominated non-dominating point is *logically* a failure: $\theta(x_k) < 0$ proves we could have done better

The derivative-free liberation:

- Without $\theta(x_k)$, the algorithm asks at every step: "is $\tilde{x}$ better than x_k?" — and in the MO sense this has *no unique answer*
- The natural response to ambiguity is: keep both, and at the next iteration *choose which archive element to poll from*
- This breaks the single-point paradigm and makes the archive idea **inevitable**

Why did this idea arise in derivative-free first?

This is a question worth pausing on.

The gradient-based bias:

- Gradient-based MO methods are natural extensions of steepest descent: compute $\theta(x_k)$, move in $d^*(x_k)$, maintain a *single-point sequence*
- This paradigm is *so natural* that the question "what if we keep multiple points?" is never forced by the mathematics
- A nondominated non-dominating point is *logically* a failure: $\theta(x_k) < 0$ proves we could have done better

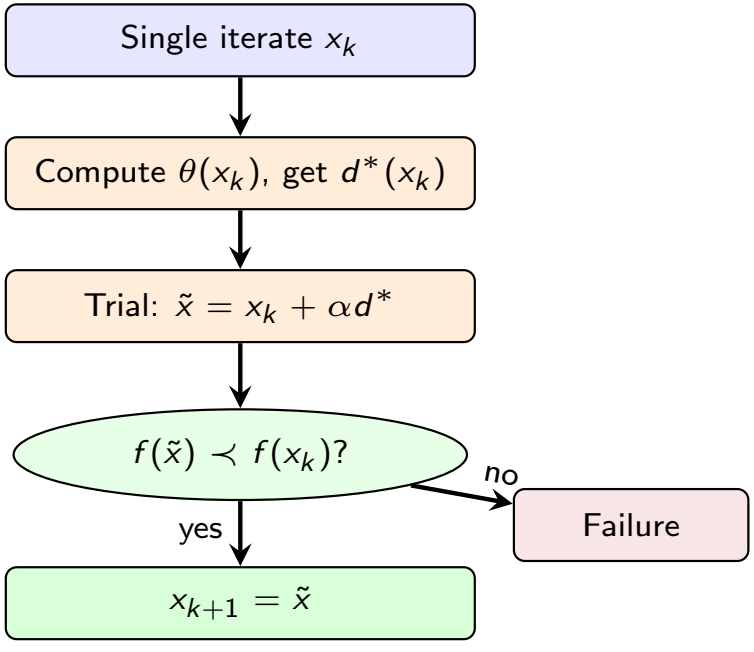
The derivative-free liberation:

- Without $\theta(x_k)$, the algorithm asks at every step: "is $\tilde{x}$ better than x_k?" — and in the MO sense this has *no unique answer*
- The natural response to ambiguity is: keep both, and at the next iteration *choose which archive element to poll from*
- This breaks the single-point paradigm and makes the archive idea **inevitable**

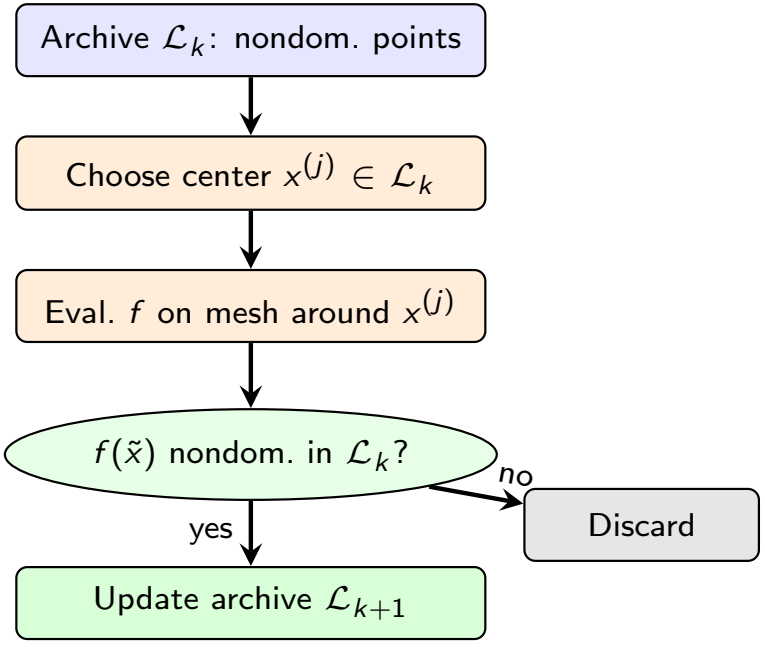
Historical note: gradient-based Pareto approximation methods (maintaining lists of iterates) came later, inspired by the derivative-free literature.

Contrast: single-point vs. archive-based

Gradient-Based



Derivative-Free (DMS)



Success = domination (left) vs. non-domination (right).

Today's plan

1. **The Direct Multisearch (DMS) algorithm**
   - Archive maintenance, polling, step-size update rule
   - The notion of sufficient decrease in the MO sense
2. **Convergence of the step-size**
   - $\alpha_k \to 0$ along unsuccessful subsequences
3. **Convergence to Pareto stationarity**
   - Limit points of archive sequences are Pareto stationary
   - Relation to the single-objective case ($p = 1$: reduces to coordinate search)

Reference: A.L. Custódio, J.F.A. Madeira, A.I.F. Vaz, L.N. Vicente, *Direct Multisearch for Multiobjective Optimization*, SIAM J. Optim., 21(3):1109–1140, 2011.

Starting the search: the first iterate

Imagine we are starting our derivative-free multiobjective algorithm. We have an initial point $x_0 \in \mathbb{R}^n$ and an initial step-size $\alpha_0 > 0$.

First step: Just like in the single-objective case, we generate a set of poll directions $\mathcal{D}_0$.

Starting the search: the first iterate

Imagine we are starting our derivative-free multiobjective algorithm. We have an initial point $x_0 \in \mathbb{R}^n$ and an initial step-size $\alpha_0 > 0$.

First step: Just like in the single-objective case, we generate a set of poll directions $\mathcal{D}_0$.

Question: What kind of directions should we use?

Starting the search: the first iterate

Imagine we are starting our derivative-free multiobjective algorithm. We have an initial point $x_0 \in \mathbb{R}^n$ and an initial step-size $\alpha_0 > 0$.

First step: Just like in the single-objective case, we generate a set of poll directions $\mathcal{D}_0$.

Question: What kind of directions should we use?

Answer: A **positive spanning set** (e.g., the coordinate directions $\pm e_i$). Why? Because if x_0 is not Pareto stationary, a common descent direction exists. A positive spanning set guarantees that at least one direction will have a positive projection on that descent direction!

Starting the search: the first iterate

Imagine we are starting our derivative-free multiobjective algorithm. We have an initial point $x_0 \in \mathbb{R}^n$ and an initial step-size $\alpha_0 > 0$.

First step: Just like in the single-objective case, we generate a set of poll directions $\mathcal{D}_0$.

Question: What kind of directions should we use?

Answer: A **positive spanning set** (e.g., the coordinate directions $\pm e_i$). Why? Because if x_0 is not Pareto stationary, a common descent direction exists. A positive spanning set guarantees that at least one direction will have a positive projection on that descent direction! **MAYBE??**

Starting the search: the first iterate

Imagine we are starting our derivative-free multiobjective algorithm. We have an initial point $x_0 \in \mathbb{R}^n$ and an initial step-size $\alpha_0 > 0$.

First step: Just like in the single-objective case, we generate a set of poll directions $\mathcal{D}_0$.

Question: What kind of directions should we use?

Answer: A **positive spanning set** (e.g., the coordinate directions $\pm e_i$). Why? Because if x_0 is not Pareto stationary, a common descent direction exists. A positive spanning set guarantees that at least one direction will have a positive projection on that descent direction! **MAYBE??**

Question: In single-objective direct search, we often use *opportunistic polling* (we evaluate trial points one by one and stop as soon as we find $f(x_0 + \alpha_0 d) < f(x_0)$). Should we do that here?

Starting the search: the first iterate

Imagine we are starting our derivative-free multiobjective algorithm. We have an initial point $x_0 \in \mathbb{R}^n$ and an initial step-size $\alpha_0 > 0$.

First step: Just like in the single-objective case, we generate a set of poll directions $\mathcal{D}_0$.

Question: What kind of directions should we use?

Answer: A **positive spanning set** (e.g., the coordinate directions $\pm e_i$). Why? Because if x_0 is not Pareto stationary, a common descent direction exists. A positive spanning set guarantees that at least one direction will have a positive projection on that descent direction! **MAYBE??**

Question: In single-objective direct search, we often use *opportunistic polling* (we evaluate trial points one by one and stop as soon as we find $f(x_0 + \alpha_0 d) < f(x_0)$). Should we do that here?

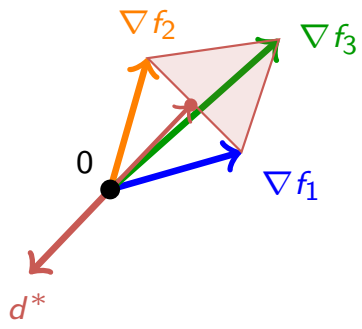
Answer: Usually, **no**. In DMS, we generally perform a **complete poll**, evaluating ALL directions in $\mathcal{D}_0$. *Reason:* Our goal is to map an entire Pareto front, not just find a single path to a minimum. Exploring all directions gives us more candidates to expand the front!

The geometry of common descent

In single-objective optimization, the set of descent directions is an entire half-space (an angle of 180 degrees). In multiobjective optimization, a direction must decrease *all* objectives simultaneously.

Recall the convex hull formulation of the stationarity measure $\theta(x)$:

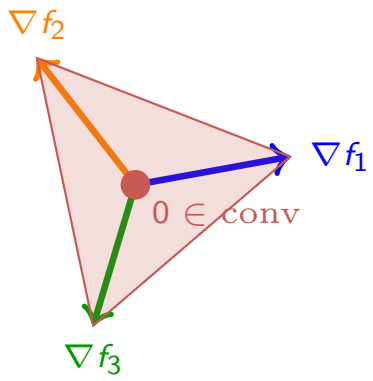
Case 1: $\theta(x) > 0$



All gradients in same half-space.

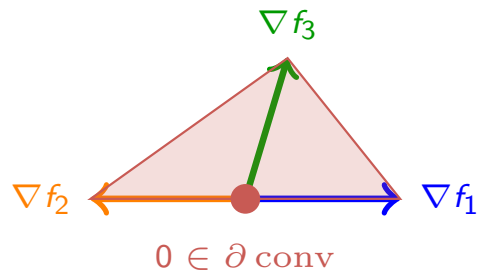
Descent direction d^* exists.

Case 2: $\theta(x) = 0$



Origin inside convex hull. Positive spanning set. x is Pareto stationary.

Case 3: $\theta(x) = 0$ (edge)



$\nabla f_1 = -\nabla f_2$: origin on boundary of convex hull. x is Pareto stationary.

The challenge of the tighter cone

As we saw, when $\theta(x) > 0$, a common descent direction exists. However, the **cone of common descent directions** can be extremely narrow!

Question: If the common descent cone is very tight, what happens when we evaluate our finite set of poll directions $\mathcal{D}_0 = \{\pm e_1, \ldots, \pm e_n\}$?

The challenge of the tighter cone

As we saw, when $\theta(x) > 0$, a common descent direction exists. However, the **cone of common descent directions** can be extremely narrow!

Question: If the common descent cone is very tight, what happens when we evaluate our finite set of poll directions $\mathcal{D}_0 = \{\pm e_1, \ldots, \pm e_n\}$?

Observation 1: It is entirely possible that **none** of our discrete poll directions fall inside this narrow cone!

This means that even if x_0 is NOT stationary, our algorithm might fail to find a point that *strictly dominates* x_0, even as the step-size $\alpha \to 0$.

The challenge of the tighter cone

As we saw, when $\theta(x) > 0$, a common descent direction exists. However, the **cone of common descent directions** can be extremely narrow!

Question: If the common descent cone is very tight, what happens when we evaluate our finite set of poll directions $\mathcal{D}_0 = \{\pm e_1, \ldots, \pm e_n\}$?

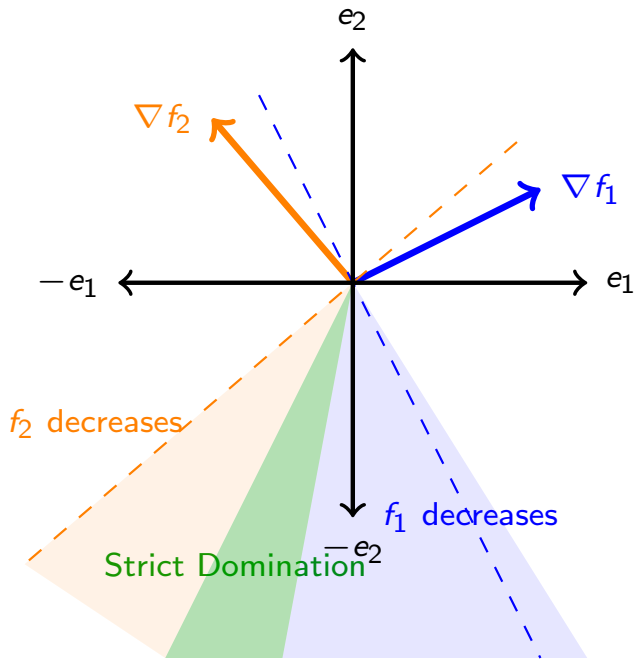
Observation 1: It is entirely possible that **none** of our discrete poll directions fall inside this narrow cone!

This means that even if x_0 is NOT stationary, our algorithm might fail to find a point that *strictly dominates* x_0, even as the step-size $\alpha \to 0$.

Question: Does this mean the iteration is a failure? Are we stuck?

Observation 2 (The Saving Grace): To find a strictly dominating point, a direction must be inside the narrow intersection of ALL half-spaces. But to find a **mutually nondominated point**, the direction only needs to fall into the descent half-space of **at least one** objective function!

Strict dominance is hard, nondominance is easy



Notice how the four coordinate directions **miss** the green cone of strict domination entirely! But $-e_2$ falls in the blue region (f_1 decreases) and $-e_1$ falls in the orange region (f_2 decreases).

Conclusion: Even when strict improvement is missed, the algorithm easily discovers new points that are **mutually nondominated** with x_0.

Managing the archive: list updates

Since we can easily generate new mutually nondominated points, we need a place to store them. We maintain an **archive** (or list) $\mathcal{L}_k$ of mutually nondominated points.

At each iteration, we evaluate our poll points, combine them with our current list, and filter out any points that are dominated.

Managing the archive: list updates

Since we can easily generate new mutually nondominated points, we need a place to store them. We maintain an **archive** (or list) $\mathcal{L}_k$ of mutually nondominated points.

At each iteration, we evaluate our poll points, combine them with our current list, and filter out any points that are dominated.

Question: Should we keep ALL the nondominated points in our new list $\mathcal{L}_{k+1}$?

Managing the archive: list updates

Since we can easily generate new mutually nondominated points, we need a place to store them. We maintain an **archive** (or list) $\mathcal{L}_k$ of mutually nondominated points.

At each iteration, we evaluate our poll points, combine them with our current list, and filter out any points that are dominated.

Question: Should we keep ALL the nondominated points in our new list $\mathcal{L}_{k+1}$?

This is a design choice with trade-offs (Custódio et al., 2011):

- **Keep all points:** Generates a rich, dense approximation of the entire Pareto front. *Drawback:* The list grows large, making iterations computationally heavier.
- **Keep a bounded list:** Limits the list to a maximum size, replacing older/crowded points. *Drawback:* Might lose some coverage of the front.
- **Keep a single point:** Reduces to a single-objective trajectory (only accept strictly dominating points). *Drawback:* Yields only one point on the Pareto front.

The Golden Rule of List Updates

Suppose we want to maintain a bounded list (e.g., maximum 5 points) to save computational time. We poll and find a 6th point that is mutually nondominated with the other 5.

Question: Can we simply throw away one of the old 5 points to make room for the 6th?

The Golden Rule of List Updates

Suppose we want to maintain a bounded list (e.g., maximum 5 points) to save computational time. We poll and find a 6th point that is mutually nondominated with the other 5.

Question: Can we simply throw away one of the old 5 points to make room for the 6th?

Answer: Absolutely NOT!

To guarantee mathematical convergence, Custódio et al. impose a strict **Retention Rule**:

> *"$\mathcal{L}_{trial}$ must necessarily include* **all** *the nondominated points that belonged to the iterate list considered at the previous iteration."*

The Golden Rule of List Updates

Suppose we want to maintain a bounded list (e.g., maximum 5 points) to save computational time. We poll and find a 6th point that is mutually nondominated with the other 5.

Question: Can we simply throw away one of the old 5 points to make room for the 6th?

Answer: Absolutely NOT!

To guarantee mathematical convergence, Custódio et al. impose a strict **Retention Rule**:

> *"$\mathcal{L}_{trial}$ must necessarily include* **all** *the nondominated points that belonged to the iterate list considered at the previous iteration."*

Why? Because of the **danger of infinite cycling**. If we casually throw away an old point A to accept a new mutually nondominated point B, we might later poll around B, rediscover A (or a point very close to it), throw away B to accept A, and circle around the same region forever!

The Single-Point Paradox

If the Retention Rule states we *cannot* throw away old nondominated points, we face a logical paradox.

Question: How can we possibly run the "single-point" variant of this algorithm? If we start with $\mathcal{L}_k = \{x_k\}$, and we find a new mutually nondominated point y, doesn't the list inevitably grow to size 2?

The Single-Point Paradox

If the Retention Rule states we *cannot* throw away old nondominated points, we face a logical paradox.

Question: How can we possibly run the "single-point" variant of this algorithm? If we start with $\mathcal{L}_k = \{x_k\}$, and we find a new mutually nondominated point y, doesn't the list inevitably grow to size 2?

Answer: The secret lies in reading the rule carefully. The rule protects **old** points from being discarded arbitrarily. It does **not** force us to accept **new** points!

The Single-Point Paradox

If the Retention Rule states we *cannot* throw away old nondominated points, we face a logical paradox.

Question: How can we possibly run the "single-point" variant of this algorithm? If we start with $\mathcal{L}_k = \{x_k\}$, and we find a new mutually nondominated point y, doesn't the list inevitably grow to size 2?

Answer: The secret lies in reading the rule carefully. The rule protects **old** points from being discarded arbitrarily. It does **not** force us to accept **new** points!

How the single-point variant works:

- **Case 1: y strictly dominates x_k.** During filtering, the old point x_k is marked as dominated. We are now legally allowed to throw it away! $\mathcal{L}_{k+1} = \{y\}$. (List size $= 1$, Success!).
- **Case 2: y is mutually nondominated with x_k.** During filtering, x_k survives. The Retention Rule forces us to keep x_k. To keep the list size at 1, we simply **choose to discard the new point** y. $\mathcal{L}_{k+1} = \{x_k\}$. (List size $= 1$, Failure!).

Preventing loops: the role of the step-size

Question: If we keep discarding these new mutually nondominated points (Case 2), won't we just poll again, find them again, and get stuck in an infinite loop of finding and discarding?

Preventing loops: the role of the step-size

Question: If we keep discarding these new mutually nondominated points (Case 2), won't we just poll again, find them again, and get stuck in an infinite loop of finding and discarding?

Answer: No! And this is where the step-size management saves the algorithm.

Preventing loops: the role of the step-size

Question: If we keep discarding these new mutually nondominated points (Case 2), won't we just poll again, find them again, and get stuck in an infinite loop of finding and discarding?

Answer: No! And this is where the step-size management saves the algorithm.

If the list does not change ($\mathcal{L}_{k+1} = \mathcal{L}_k$), the iteration is declared **unsuccessful**.

Whenever an iteration is unsuccessful, the algorithm strictly **shrinks the step-size** associated with that poll center:

$$\alpha_{k+1} = \theta \alpha_k \quad \text{with} \quad \theta \in (0, 1).$$

Preventing loops: the role of the step-size

Question: If we keep discarding these new mutually nondominated points (Case 2), won't we just poll again, find them again, and get stuck in an infinite loop of finding and discarding?

Answer: No! And this is where the step-size management saves the algorithm.

If the list does not change ($\mathcal{L}_{k+1} = \mathcal{L}_k$), the iteration is declared **unsuccessful**.

Whenever an iteration is unsuccessful, the algorithm strictly **shrinks the step-size** associated with that poll center:

$$\alpha_{k+1} = \theta \alpha_k \quad \text{with} \quad \theta \in (0, 1).$$

Consequences:

1. **We change the search radius:** In the very next iteration, we will look at trial points *closer* to x_k. We will not evaluate the exact same discarded points again.
2. **We force convergence:** Infinite failures mean the step-size α_k shrinks geometrically toward zero. Once $\alpha_k < \alpha_{\text{min}}$, the algorithm terminates and successfully declares x_k as a Pareto stationary point.

Summary of the logic

The beauty of derivative-free convergence theory lies in how these rules lock together to force progress:

- **Strict Monotonicity:** By only ever discarding old points when they are *strictly dominated*, the "Pareto quality" of the archive strictly improves. We never take a step backward.
- **Forced Shrinkage:** If we cannot expand the frontier or strictly improve our points, the iteration fails.
- **Step-size Convergence:** Repeated failures geometrically shrink $\alpha_k \to 0$, forcing the search to become microscopic and triggering termination.
- **Stationarity Guarantee:** If $\alpha_k \to 0$, it is a mathematical certainty that no direction in our positive spanning set could yield a common descent, proving we have reached a Pareto stationary point! **OR HAVE WE NOT??**

Choosing the next poll center

Suppose we maintain a list of multiple nondominated points. Our algorithm is iterative: at step $k+1$, we must choose a point from $\mathcal{L}_{k+1}$ around which to poll.

Question: How do we select the next poll center?

Choosing the next poll center

Suppose we maintain a list of multiple nondominated points. Our algorithm is iterative: at step $k+1$, we must choose a point from $\mathcal{L}_{k+1}$ around which to poll.

Question: How do we select the next poll center?

Answer: The mathematical convergence of the algorithm **does not depend** on how you select the point, as long as you systematically visit them!

However, in practice, this choice determines how well you explore the objective space:

- **FIFO (Queue):** Cycle through the list sequentially. Guarantees every point gets polled, ensuring a uniform spread.
- **Spread metrics:** Choose points that are in the most "empty" regions of the current Pareto front approximation to encourage uniform distribution.
- **Most recently added (Stack):** Focuses on deepening a specific region of the front quickly.

Managing the step-size

We have a list of points, and we are jumping between different poll centers at each iteration.

Question: In single-objective direct search, we have one step-size α_k. Should we use a single, global step-size for all points in the multiobjective list?

Managing the step-size

We have a list of points, and we are jumping between different poll centers at each iteration.

Question: In single-objective direct search, we have one step-size α_k. Should we use a single, global step-size for all points in the multiobjective list?

Let's think about it: Suppose point A is in a very "flat" region where large steps are still successful. Point B is trapped near a sharp Pareto optimal corner where all large steps fail.

If we use a global step-size, polling at B will cause α to shrink. When we switch back to polling at A, the step-size will be unnecessarily small, ruining our efficiency!

Managing the step-size

We have a list of points, and we are jumping between different poll centers at each iteration.

Question: In single-objective direct search, we have one step-size α_k. Should we use a single, global step-size for all points in the multiobjective list?

Let's think about it: Suppose point A is in a very "flat" region where large steps are still successful. Point B is trapped near a sharp Pareto optimal corner where all large steps fail.

If we use a global step-size, polling at B will cause α to shrink. When we switch back to polling at A, the step-size will be unnecessarily small, ruining our efficiency!

The Solution: Local step-sizes. We do not just store points in our list; we store pairs: (x, α). Every point carries its own step-size parameter!

- When we poll around x and succeed, we add the new points to the list with step-size α (or expanded).
- When we poll around x and fail, we shrink its specific α, leaving the step-sizes of other points untouched.

Putting it all together: Direct Multisearch (DMS)

Input: $x_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\alpha_{\min} > 0$, $\theta \in (0,1)$

Initialize list $\mathcal{L}_0 = \{(x_0, \alpha_0)\}$, $k \leftarrow 0$
while $\max_{(x,\alpha)\in\mathcal{L}_k} \alpha \geq \alpha_{\min}$ **do**
  Select a poll center (x_k, α_k) from $\mathcal{L}_k$
  Generate a positive spanning set $\mathcal{D}_k$
  Poll: Evaluate $f(x_k + \alpha_k d)$ for all $d \in \mathcal{D}_k$
  Filter: Extract mutually nondominated points from $\mathcal{L}_k \cup \{\text{new valid trial points}\}$
  Update: Select the trial list $\mathcal{L}_{trial}$ from these filtered points
  if $\mathcal{L}_{trial} \neq \mathcal{L}_k$ **then**
    Success: $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_{trial}$
    Step-size: Assign $\alpha_{new} \geq \alpha_k$ to the newly added points
  else
    Failure: $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k$
    Step-size: Shrink α ONLY for the center x_k. Replace (x_k, α_k) with $(x_k, \theta\alpha_k)$
  end if
  $k \leftarrow k + 1$
end while
return $\mathcal{L}_k$

Putting it all together: Direct Multisearch (DMS)

Input: $x_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\alpha_{\min} > 0$, $\theta \in (0,1)$

1: **Initialize** list $\mathcal{L}_0 = \{(x_0, \alpha_0)\}$, $k \leftarrow 0$
2: **while** $\max_{(x,\alpha)\in\mathcal{L}_k} \alpha \geq \alpha_{\min}$ **do**
3: **Select** a poll center (x_k, α_k) from $\mathcal{L}_k$
4: **Generate** a positive spanning set $\mathcal{D}_k$
5: **Poll**: Evaluate $f(x_k + \alpha_k d)$ for all $d \in \mathcal{D}_k$
6: **Filter**: Extract mutually nondominated points from $\mathcal{L}_k \cup \{\text{new valid trial points}\}$
7: **Update**: Select the trial list $\mathcal{L}_{trial}$ from these filtered points
8: **if** $\mathcal{L}_{trial} \neq \mathcal{L}_k$ **then**
9: **Success**: $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_{trial}$
10: *Step-size*: Assign $\alpha_{new} \geq \alpha_k$ to the newly added points
11: **else**
12: **Failure**: $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k$
13: *Step-size*: Shrink α ONLY for the center x_k. Replace (x_k, α_k) with $(x_k, \theta\alpha_k)$
14: **end if**
15: $k \leftarrow k + 1$
16: **end while**
17: **return** $\mathcal{L}_k$

Question: Looking at the strict '$\neq$' condition in line 7, what is the exact definition of

Numerical Example: Problem Setup

Let's trace the algorithm on a simple unconstrained problem with $p = 2$ objectives:

$$F(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} = \begin{pmatrix} (x_1 - 1)^2 + (x_1 - x_2)^2 \\ (x_1 - x_2)^2 + (x_2 - 3)^2 \end{pmatrix}$$

Initialization:

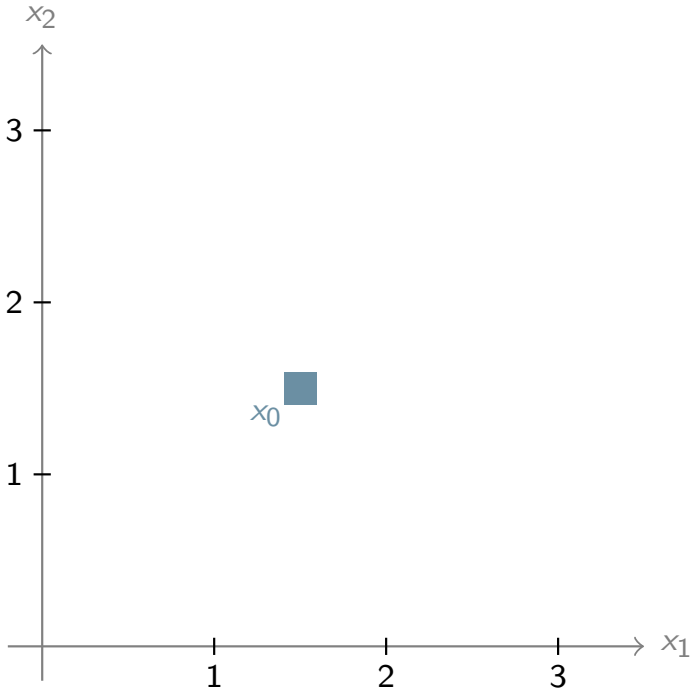
- Starting point $x_0 = (1.5, 1.5) \implies F(x_0) = (0.25, 2.25)$
- Initial list $\mathcal{L}_0 = \{(x_0, 1)\}$
- Step-size $\alpha = 1$, Poll directions $\mathcal{D} = \{\pm e_1, \pm e_2\}$

In the following slides, we will observe the algorithm operating simultaneously in the **Decision Space** *(where we generate trial points) and the* **Objective Space** *(where we filter them).*

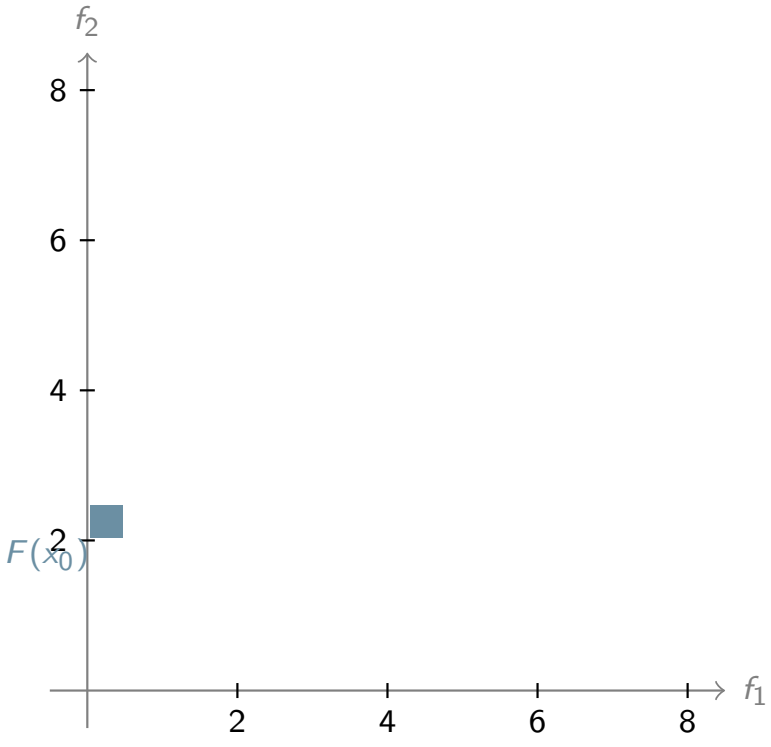
Iteration 0: Polling around x_0

Step 1: Select x_0. Evaluate the 4 poll points $x_0 \pm \alpha e_i$.

Decision Space (x_1, x_2)



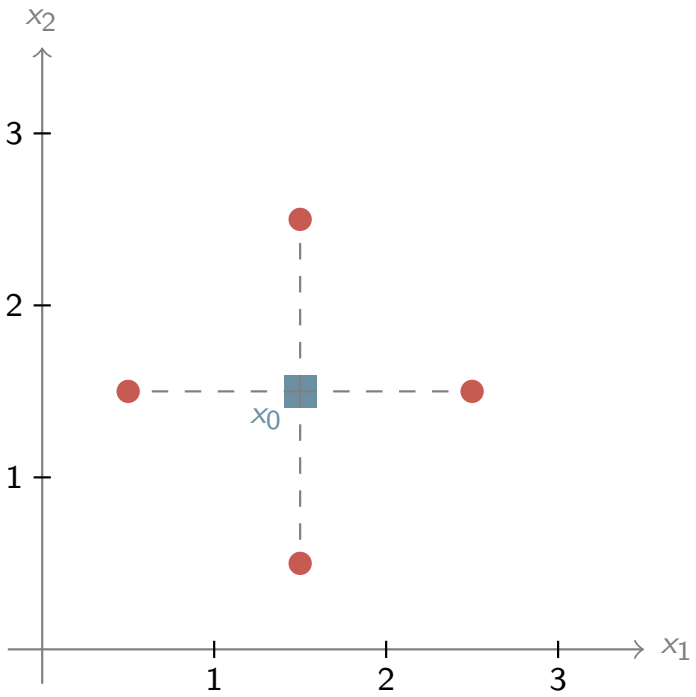
Objective Space (f_1, f_2)



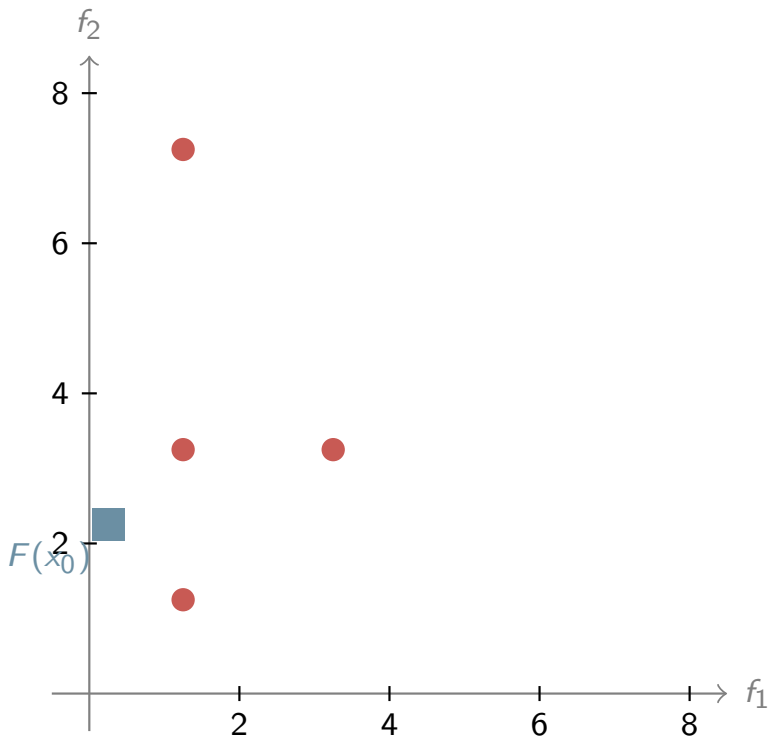
Iteration 0: Polling around x_0

Step 2: Map the points to the objective space. Notice how $x_0 + e_2$ moves closer to the origin!

Decision Space (x_1, x_2)



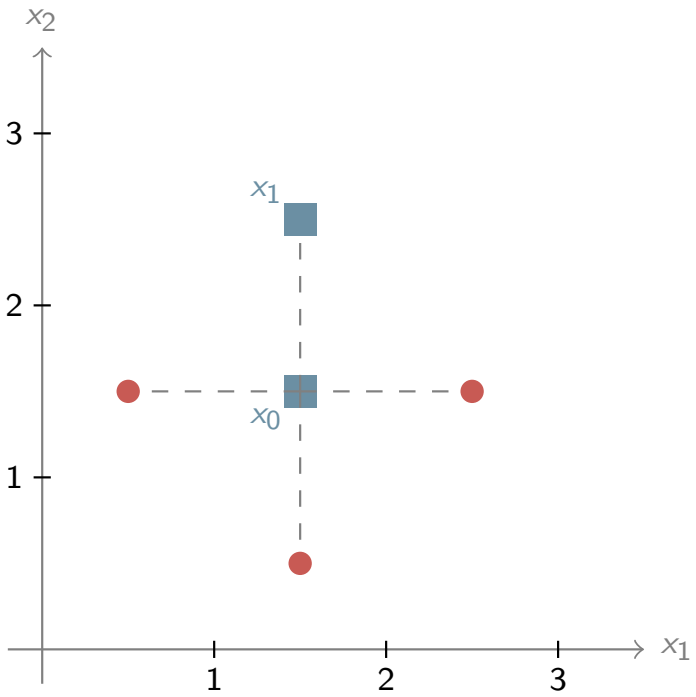
Objective Space (f_1, f_2)



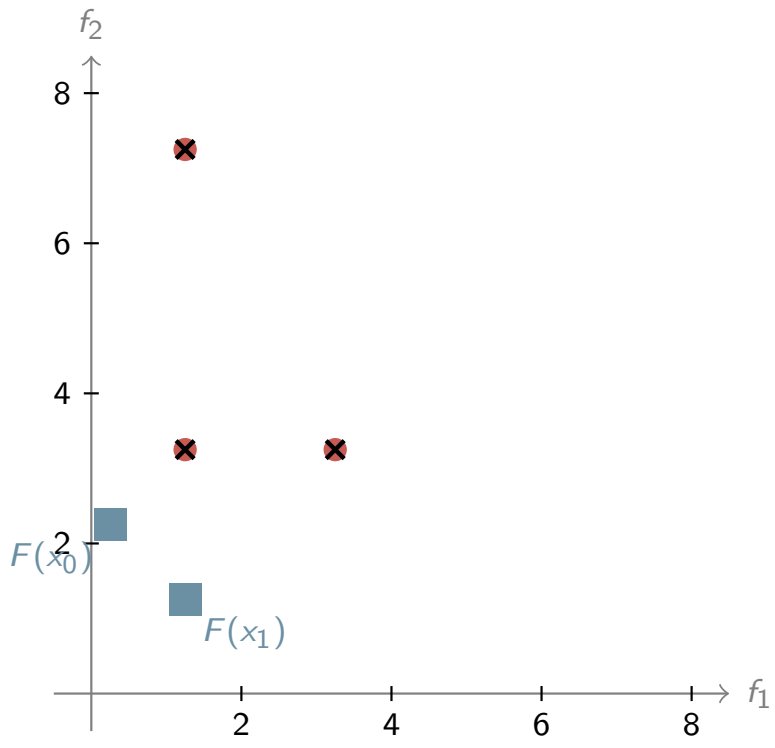
Iteration 0: Polling around x_0

Step 3: Filter. Cross out dominated points. The new point is mutually nondominated with x_0. Add it to the list to form $\mathcal{L}_1$!

Decision Space (x_1, x_2)



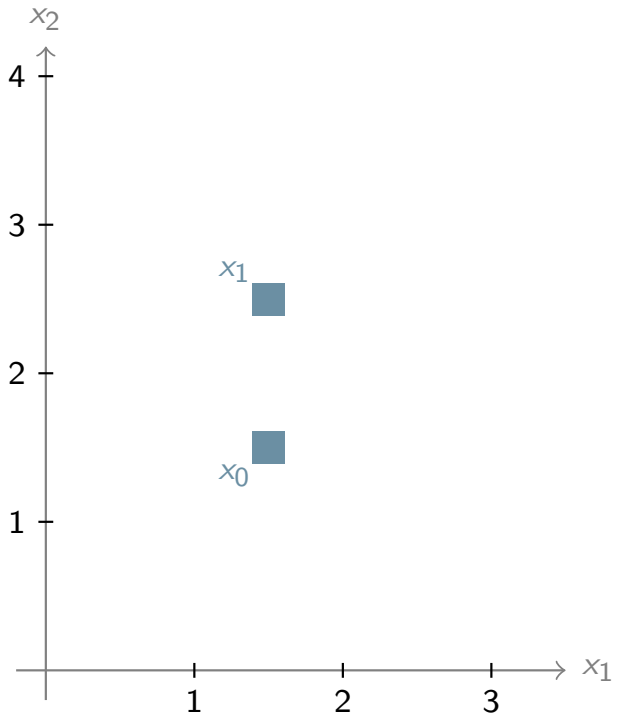
Objective Space (f_1, f_2)



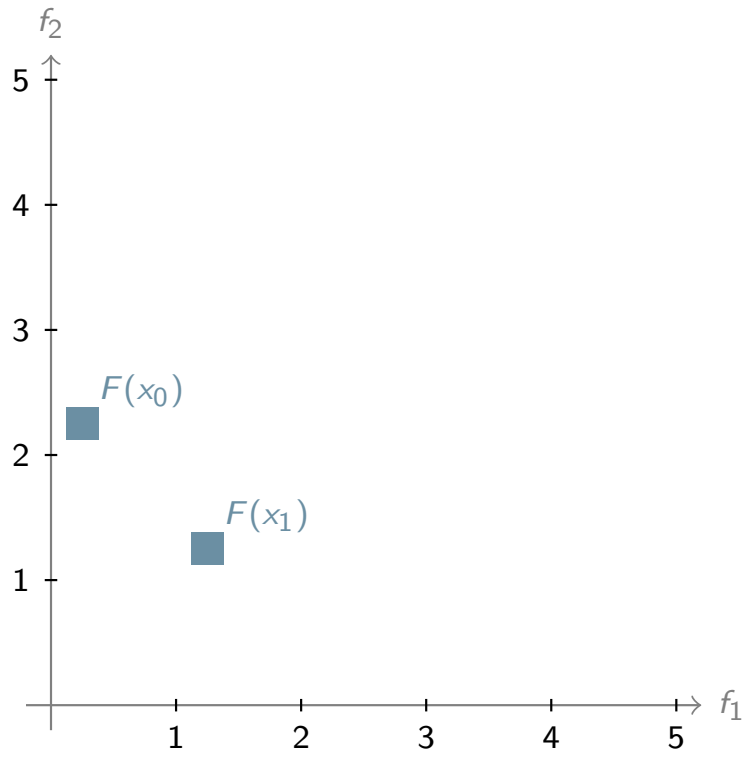
Iteration 1: Polling around x_1

Step 1: Select the newly added $x_1 = (1.5, 2.5)$ as our next poll center.

Decision Space (x_1, x_2)



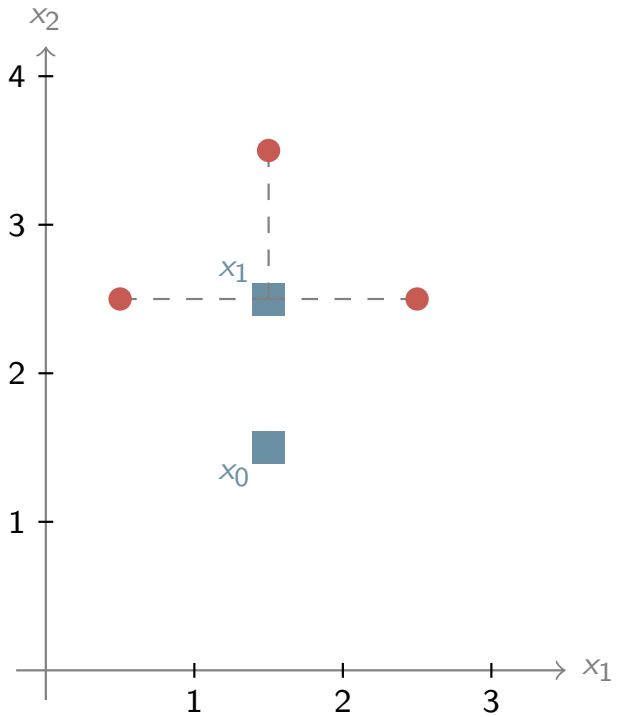
Objective Space (f_1, f_2)



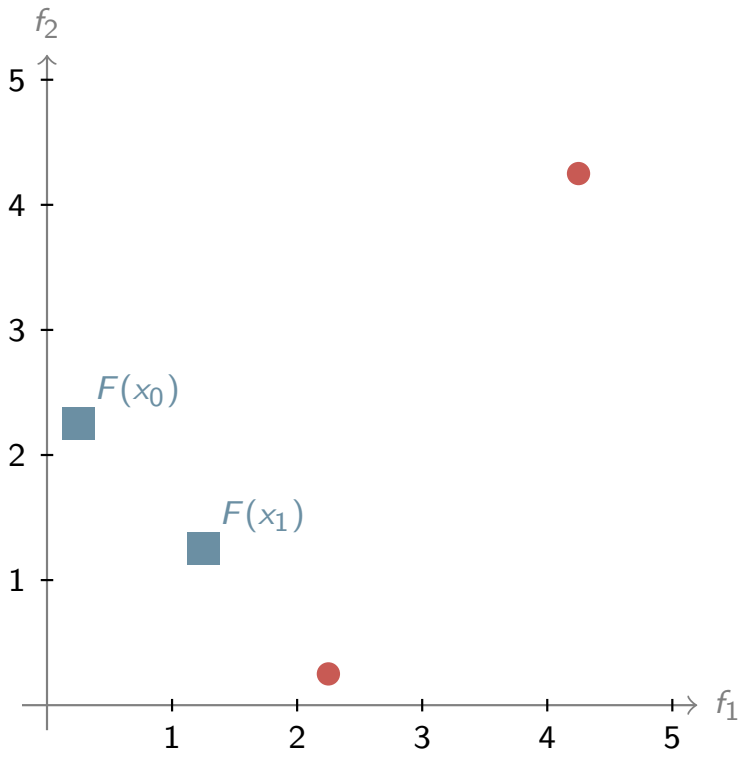
Iteration 1: Polling around x_1

Step 2: Evaluate $x_1 \pm e_i$. (Note: $x_1 - e_2$ is just x_0, we don't need to re-evaluate it).

Decision Space (x_1, x_2)



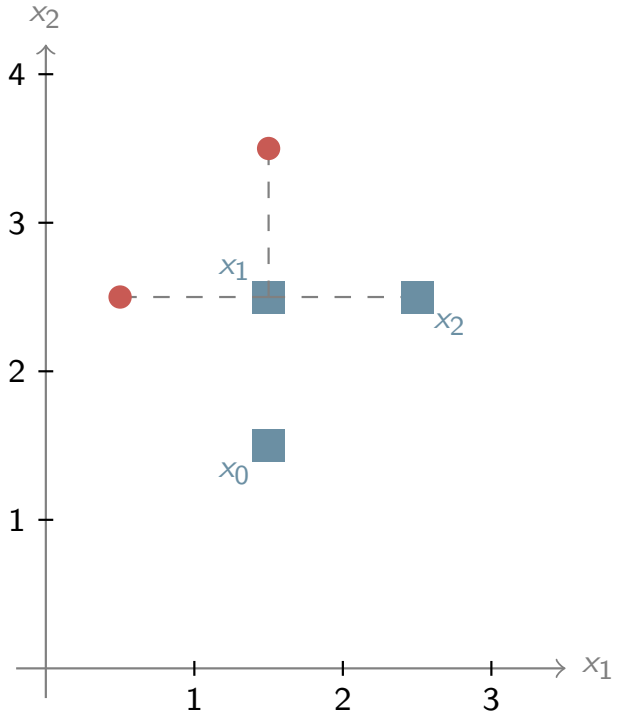
Objective Space (f_1, f_2)



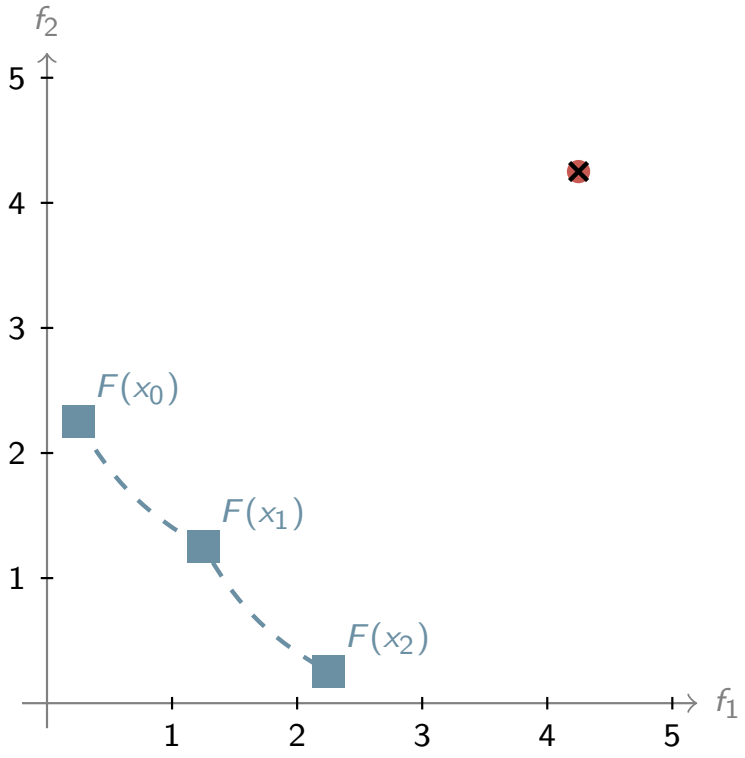
Iteration 1: Polling around x_1

Step 3: Filter. Two points are heavily dominated, but $x_1 + e_1$ pushes the frontier forward! The Pareto front emerges.

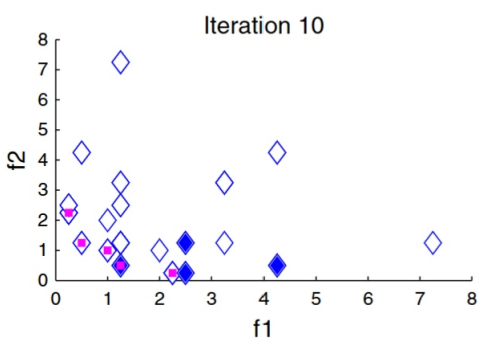
Decision Space (x_1, x_2)

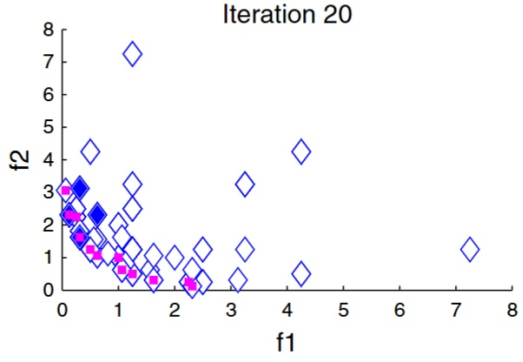


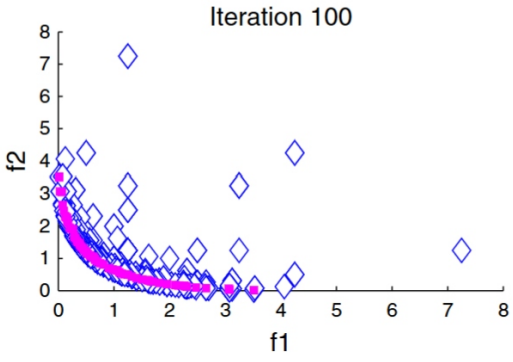
Objective Space (f_1, f_2)



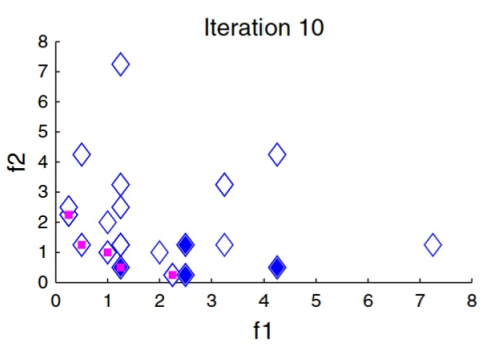
Long-term behavior of DMS

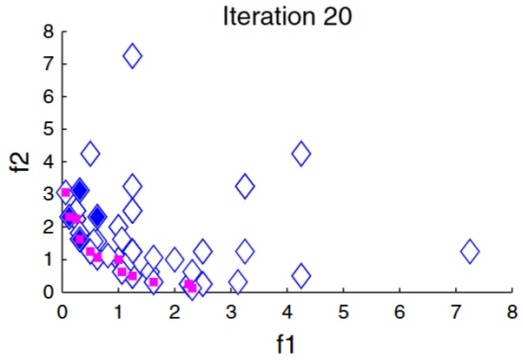


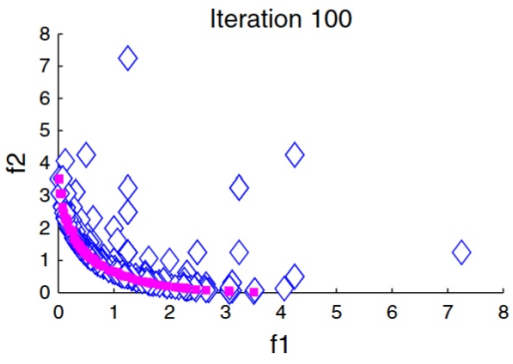




Long-term behavior of DMS







Key Observations:

- **Exploration:** By cycling through different poll centers in the list, the algorithm extensively explores and maps the entire objective space rather than getting stuck in one region.
- **Front Expansion:** The list of nondominated points (magenta squares) steadily grows in size and pushes deeper towards the bottom-left ideal point.
- **Refinement:** As the algorithm gets closer to the true Pareto front, large steps begin to fail. Unsuccessful iterations shrink the step-sizes, naturally leading to a fine-grained, high-resolution approximation of the continuous curve at iteration 100.

Globalization strategies: a recap

For the algorithm to converge, we need a mechanism that forces the step-size α_k to shrink to zero.

Recall from the single-objective direct search the two main **globalization strategies**:

1. **Integer Lattice (Discrete Mesh):** Points are evaluated on a fixed grid that only shrinks on failures. Because the level set is compact, the grid contains finitely many points, forcing failures eventually.

2. **Sufficient Decrease:** We reject steps that are "too small" by enforcing a forcing function $\rho(\alpha) = \gamma\alpha^2$. We only accept a step if it strictly improves the objective by at least $\rho(\alpha_k)$.

Globalization strategies: a recap

For the algorithm to converge, we need a mechanism that forces the step-size α_k to shrink to zero.

Recall from the single-objective direct search the two main **globalization strategies**:

1. **Integer Lattice (Discrete Mesh):** Points are evaluated on a fixed grid that only shrinks on failures. Because the level set is compact, the grid contains finitely many points, forcing failures eventually.

2. **Sufficient Decrease:** We reject steps that are "too small" by enforcing a forcing function $\rho(\alpha) = \gamma\alpha^2$. We only accept a step if it strictly improves the objective by at least $\rho(\alpha_k)$.

Question: In single-objective, sufficient decrease is simply $f(x_{new}) \leq f(x_k) - \rho(\alpha_k)$. How does a sufficient decrease look like in the multiobjective case?

Multiobjective sufficient decrease

In multiobjective optimization, we evaluate a trial point against an entire list $\mathcal{L}_k$ of nondominated points.

If we just ask for standard nondominance, a new point x_{new} might be accepted just because it has $f_1(x_{new}) = f_1(y) - 10^{-10}$. For a rigorous convergence, we must demand a **meaningful** improvement!

Multiobjective sufficient decrease

In multiobjective optimization, we evaluate a trial point against an entire list $\mathcal{L}_k$ of nondominated points.

If we just ask for standard nondominance, a new point x_{new} might be accepted just because it has $f_1(x_{new}) = f_1(y) - 10^{-10}$. For a rigorous convergence, we must demand a **meaningful** improvement!

The Multiobjective Sufficient Decrease Condition:

A trial point x_{new} provides a sufficient decrease if it decreases **at least one objective function** of **at least one point** currently in the list by the forcing amount $\rho(\alpha_k)$.

Multiobjective sufficient decrease

In multiobjective optimization, we evaluate a trial point against an entire list $\mathcal{L}_k$ of nondominated points.

If we just ask for standard nondominance, a new point x_{new} might be accepted just because it has $f_1(x_{new}) = f_1(y) - 10^{-10}$. For a rigorous convergence, we must demand a **meaningful** improvement!

The Multiobjective Sufficient Decrease Condition:

A trial point x_{new} provides a sufficient decrease if it decreases **at least one objective function** of **at least one point** currently in the list by the forcing amount $\rho(\alpha_k)$.

Mathematically, x_{new} is accepted if there exists $y \in \mathcal{L}_k$ and an index $j \in \{1, \ldots, p\}$ such that:

$$f_j(x_{new}) \leq f_j(y) - \rho(\alpha_k)$$

Note: x_{new} doesn't have to be better in all objectives, and it doesn't even have to dominate y. It just has to push the Pareto front approximation forward by at least $\rho(\alpha_k)$ in some direction!

Assumptions

(A1) **Compactness**: The combined sub-level set

$$\mathcal{L}(x_0) = \bigcup_{i=1}^{p} \{x \in \mathbb{R}^n : f_i(x) \leq f_i(x_0)\}$$

is compact. (The iterates cannot escape to infinity).

(A2) **Smoothness**: All objective components f_i are continuously differentiable.

Step-size convergence

Let's prove that this sufficient decrease mechanism actually forces $\alpha_k \to 0$.

Theorem

Let the combined level set $\mathcal{L}(x_0)$ be compact (Assumption A1). If DMS uses the sufficient decrease condition with a forcing function $\rho(\alpha) > 0$ for $\alpha > 0$, then:

$$\liminf_{k \to \infty} \alpha_k = 0$$

Proof Strategy: Contradiction.

Assume the step-size does **not** converge to zero. This means there is a strictly positive lower bound: $\alpha_k \geq \bar{\alpha} > 0$ for all k.

Step-size convergence

Let's prove that this sufficient decrease mechanism actually forces $\alpha_k \to 0$.

Theorem

Let the combined level set $\mathcal{L}(x_0)$ be compact (Assumption A1). If DMS uses the sufficient decrease condition with a forcing function $\rho(\alpha) > 0$ for $\alpha > 0$, then:

$$\liminf_{k \to \infty} \alpha_k = 0$$

Proof Strategy: Contradiction.

Assume the step-size does **not** converge to zero. This means there is a strictly positive lower bound: $\alpha_k \geq \bar{\alpha} > 0$ for all k.

If $\alpha_k \geq \bar{\alpha}$, then the required decrease is also bounded away from zero:

$$\rho(\alpha_k) \geq \rho(\bar{\alpha}) = \bar{\rho} > 0$$

Step-size convergence

Let's prove that this sufficient decrease mechanism actually forces $\alpha_k \to 0$.

Theorem

Let the combined level set $\mathcal{L}(x_0)$ be compact (Assumption A1). If DMS uses the sufficient decrease condition with a forcing function $\rho(\alpha) > 0$ for $\alpha > 0$, then:

$$\liminf_{k \to \infty} \alpha_k = 0$$

Proof Strategy: Contradiction.
Assume the step-size does **not** converge to zero. This means there is a strictly positive lower bound: $\alpha_k \geq \bar{\alpha} > 0$ for all k.

If $\alpha_k \geq \bar{\alpha}$, then the required decrease is also bounded away from zero:

$$\rho(\alpha_k) \geq \rho(\bar{\alpha}) = \bar{\rho} > 0$$

Since the step-size is reduced by $\theta \in (0, 1)$ at every unsuccessful iteration, we can only have a *finite* number of failures. Therefore, the algorithm must undergo an **infinite number of successful iterations**.

The proof: single vs. multiobjective

We have established that the algorithm must add an infinite number of points, each yielding a decrease of at least $\bar{\rho} > 0$.

Single-Objective Case

At each success, the *single* objective function drops by $\bar{\rho}$:

$$f(x_{k+1}) \leq f(x_k) - \bar{\rho}$$

After an infinite number of successes, $f \to -\infty$.

This contradicts the fact that continuous functions are bounded below on a compact level set. ✓

The proof: single vs. multiobjective

We have established that the algorithm must add an infinite number of points, each yielding a decrease of at least $\bar{\rho} > 0$.

Single-Objective Case

At each success, the *single* objective function drops by $\bar{\rho}$:

$$f(x_{k+1}) \leq f(x_k) - \bar{\rho}$$

After an infinite number of successes, $f \to -\infty$.

This contradicts the fact that continuous functions are bounded below on a compact level set. ✓

Multiobjective Case

At each success, a new point drops *some* objective j relative to *some* point $y \in \mathcal{L}_k$ by $\bar{\rho}$:

$$f_j(x_{new}) \leq f_j(y) - \bar{\rho}$$

But the incumbent is a **list** that changes! How do we track the decrease?

The proof: single vs. multiobjective

We have established that the algorithm must add an infinite number of points, each yielding a decrease of at least $\bar{\rho} > 0$.

Single-Objective Case

At each success, the *single* objective function drops by $\bar{\rho}$:

$$f(x_{k+1}) \leq f(x_k) - \bar{\rho}$$

After an infinite number of successes, $f \to -\infty$.

This contradicts the fact that continuous functions are bounded below on a compact level set. ✓

Multiobjective Case

At each success, a new point drops *some* objective j relative to *some* point $y \in \mathcal{L}_k$ by $\bar{\rho}$:

$$f_j(x_{new}) \leq f_j(y) - \bar{\rho}$$

But the incumbent is a **list** that changes! How do we track the decrease?

Question: If a point y is later removed from the list, what happens to the objective values?

It is removed **only** because a new point strictly dominates it! Thus, the objective values in the list only get smaller, never larger.

The proof: the pigeonhole argument

So, every time a new point is added, at least one of the p objective functions is pushed down by $\bar{\rho} > 0$.

Since we have an infinite number of successful additions, and only a **finite number of objective functions** (p), the Pigeonhole Principle guarantees that:

At least one specific objective component f_j must be decreased by $\bar{\rho}$ an infinite number of times.

The proof: the pigeonhole argument

So, every time a new point is added, at least one of the p objective functions is pushed down by $\bar{\rho} > 0$.

Since we have an infinite number of successful additions, and only a **finite number of objective functions** (p), the Pigeonhole Principle guarantees that:

At least one specific objective component f_j must be decreased by $\bar{\rho}$ an infinite number of times.

Therefore, that specific objective function must diverge:

$$f_j \to -\infty$$

But the combined level set $\mathcal{L}(x_0)$ is compact, meaning **all** objective components are bounded below. We have reached our contradiction!

The proof: the pigeonhole argument

So, every time a new point is added, at least one of the p objective functions is pushed down by $\bar{\rho} > 0$.

Since we have an infinite number of successful additions, and only a **finite number of objective functions** (p), the Pigeonhole Principle guarantees that:

At least one specific objective component f_j must be decreased by $\bar{\rho}$ an infinite number of times.

Therefore, that specific objective function must diverge:

$$f_j \to -\infty$$

But the combined level set $\mathcal{L}(x_0)$ is compact, meaning **all** objective components are bounded below. We have reached our contradiction!

Conclusion: Our assumption was false. The number of successful iterations must be finite, and therefore the algorithm must eventually suffer infinitely many unsuccessful iterations, forcing:

$$\liminf_{k \to \infty} \alpha_k = 0 \square$$

The final step: proving stationarity

We have proved that along a subsequence of unsuccessful iterations $\mathcal{K}$, the step-size shrinks: $\lim_{k \in \mathcal{K}} \alpha_k = 0$. Let $x_k \to \bar{x}$.

Because these iterations are unsuccessful, the trial points $x_k + \alpha_k d$ failed to provide a sufficient decrease. By passing to the limit (using the Mean Value Theorem and continuous differentiability), we obtain:

$$\forall d \in \mathcal{D}_k, \quad \exists j \in \{1, \ldots, p\} \text{ such that } \nabla f_j(\bar{x})^\top d \geq 0$$

The final step: proving stationarity

We have proved that along a subsequence of unsuccessful iterations $\mathcal{K}$, the step-size shrinks: $\lim_{k \in \mathcal{K}} \alpha_k = 0$. Let $x_k \to \bar{x}$.

Because these iterations are unsuccessful, the trial points $x_k + \alpha_k d$ failed to provide a sufficient decrease. By passing to the limit (using the Mean Value Theorem and continuous differentiability), we obtain:

$$\forall d \in \mathcal{D}_k, \quad \exists j \in \{1, \ldots, p\} \text{ such that } \nabla f_j(\bar{x})^\top d \geq 0$$

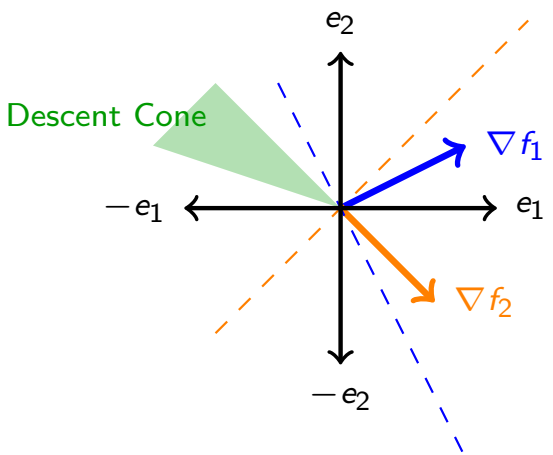
Question: In single-objective optimization ($p = 1$), if $\nabla f(\bar{x})^\top d \geq 0$ for all directions in a positive spanning set, we conclude $\nabla f(\bar{x}) = 0$.

Does this logic hold for the multiobjective case? If we use the fixed coordinate directions $\mathcal{D} = \{\pm e_1, \ldots, \pm e_n\}$ and find no common descent among them, does it guarantee that $\bar{x}$ is Pareto stationary ($\theta(\bar{x}) = 0$)?

The trap of the arbitrarily narrow cone

Answer: NO! A fixed positive spanning set is **not** enough.

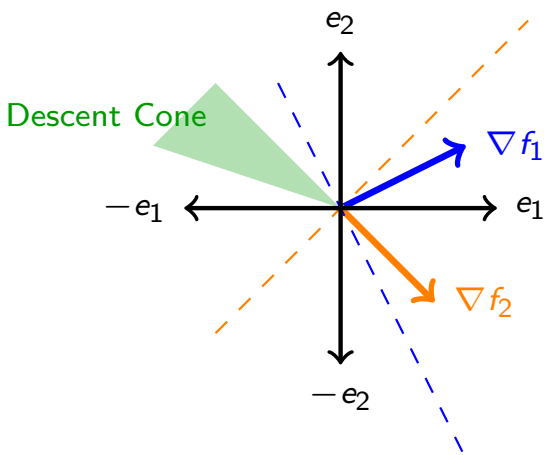
Recall our geometric intuition: if a point is not Pareto stationary ($\theta(x) > 0$), there exists a cone of common descent directions. However, this cone is the intersection of several half-spaces ($\nabla f_i(x)^\top d < 0$), and it can be **arbitrarily narrow**!



The trap of the arbitrarily narrow cone

Answer: NO! A fixed positive spanning set is **not** enough.

Recall our geometric intuition: if a point is not Pareto stationary ($\theta(x) > 0$), there exists a cone of common descent directions. However, this cone is the intersection of several half-spaces ($\nabla f_i(x)^\top d < 0$), and it can be **arbitrarily narrow**!



In the image above, the green cone of common descent clearly exists (so $\bar{x}$ is not stationary). But **none** of the coordinate directions $\pm e_i$ fall inside it! If we only poll with a fixed set, we might miss the descent cone entirely, shrink $\alpha_k \to 0$, and falsely conclude we have reached a stationary point.

The solution: density of directions

Question: If any fixed, finite set of directions might miss the cone, what must our algorithm do?

The solution: density of directions

Question: If any fixed, finite set of directions might miss the cone, what must our algorithm do?

Answer: We cannot use the exact same directions at every iteration!

We must systematically **rotate** or generate new positive spanning sets such that, over the course of the infinite run, the union of all tried directions becomes **dense** in the unit sphere.

The solution: density of directions

Question: If any fixed, finite set of directions might miss the cone, what must our algorithm do?

Answer: We cannot use the exact same directions at every iteration!

We must systematically **rotate** or generate new positive spanning sets such that, over the course of the infinite run, the union of all tried directions becomes **dense** in the unit sphere.

If the set of all evaluated directions $\bigcup_k \mathcal{D}_k$ is dense, then no matter how narrow the green cone of common descent is, eventually, at some iteration k, the algorithm will generate a direction that falls perfectly inside it!

The solution: density of directions

Question: If any fixed, finite set of directions might miss the cone, what must our algorithm do?

Answer: We cannot use the exact same directions at every iteration!

We must systematically **rotate** or generate new positive spanning sets such that, over the course of the infinite run, the union of all tried directions becomes **dense** in the unit sphere.

If the set of all evaluated directions $\bigcup_k \mathcal{D}_k$ is dense, then no matter how narrow the green cone of common descent is, eventually, at some iteration k, the algorithm will generate a direction that falls perfectly inside it!

Theorem (Pareto Stationarity)

If the sequence of directions used at unsuccessful iterations is dense in the unit sphere, then any limit point $\bar{x}$ of the refining subsequence satisfies $\theta(\bar{x}) = 0$ (it is a Pareto stationary point).

A familiar requirement: connections to nonsmoothness

If $f(x)$ is nondifferentiable, the set of descent directions might not form a neat 180-degree half-space. The "valleys" of descent around a sharp kink can be incredibly narrow.

To guarantee convergence on nonsmooth functions, single-objective direct search algorithms (like MADS) also abandon fixed coordinate searches and enforce the generation of a **dense set of directions**.

A familiar requirement: connections to nonsmoothness

If $f(x)$ is nondifferentiable, the set of descent directions might not form a neat 180-degree half-space. The "valleys" of descent around a sharp kink can be incredibly narrow.

To guarantee convergence on nonsmooth functions, single-objective direct search algorithms (like MADS) also abandon fixed coordinate searches and enforce the generation of a **dense set of directions**.

A familiar requirement: connections to nonsmoothness

If $f(x)$ is nondifferentiable, the set of descent directions might not form a neat 180-degree half-space. The "valleys" of descent around a sharp kink can be incredibly narrow.

To guarantee convergence on nonsmooth functions, single-objective direct search algorithms (like MADS) also abandon fixed coordinate searches and enforce the generation of a **dense set of directions**.

The Deep Connection:

- **Multiobjective smooth:** Narrow descent cone due to the intersection of multiple gradient half-spaces.
- **Single-objective nonsmooth:** Narrow descent cone due to the complex Clarke subdifferential at a kink.

In both cases, the algorithmic cure is the same: **directional density**.