# Optimization using Function Values Only

by

R. Simon Fong

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Thomas F. Coleman

Waterloo, Ontario, Canada, 2014

## Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

In this paper, we focus on solving global optimization problems using only the values of the objective function. In particular we explore the *Simulated Annealing* method, and find improvements using techniques in *Derivative Free Optimization* methods.

The first part of the paper examines the well known Simulated Annealing algorithm. We discuss important theoretical results, and we then demonstrate how each component of Simulated Annealing contributed to solving the global optimization problem invoking only the objective function values.

The second part of the paper will be dedicated to techniques from Derivative Free Trust Region method. We discuss how Derivative Free Trust Region determines a second order local optimum. We then propose a method to bypass local optima using Simulated Annealing hill climbing moves.

Lastly, we address the shortcomings of Simulated Annealing on continuous optimization problems with strong non-linearity. We present a new method which take into consideration topological information to determine search direction and neighborhood function.

We then look at a real life application of the Simulated Annealing method in Appendix A.

**Acknowledgements**

I would like to thank my supervisor Professor Thomas F. Coleman for his continual guidance and mentor-ship. I would also like to thank Professor Justin W.L. Wan for his insightful suggestions.

I would also like to thank my labmates for the sleepless nights we've endured, and for making this year a truly unforgettable experience.

Last but not least, I would also like to thank my family for their encouragement, support and love.

## Dedication

To my family, friends, and whoever reads this paper.

# Contents

# Chapter 1

# Introduction

## 1.1 Preliminary

*Global optimization problems* are generally expressed in the following form:

$$\begin{aligned} &\min\ f(x) \\ &\text{such that } x \in \Omega \end{aligned}$$

where $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ is called the *objective function*, and the space $\Omega$ is called the *feasible region* or *the set of feasible solutions.*

In this paper we will assume $\Omega \subseteq \mathbb{R}^n$. The problem is called *unconstrained* if $\Omega = \mathbb{R}^n$, and *constrained* if $\Omega \neq \mathbb{R}^n$.

We shall explore methods to solve global optimization problems with the objective function value only.

## 1.2 Motivation

Global Optimization problems have been a field of great interest; primarily due to their wide range of applications spanning across various fields. Many modern day problems in physics, biology, engineering, and in industries such as container terminals courier service, etc require extensive use of optimization techniques. New methods has arisen to take advantage of the advancement in computer processing power.

Classical optimization theory, using calculus and convex analysis, provides us with ways to thoroughly categorize and determine solutions to optimization problems.

The gradient of the objective function, in particular, remains one of the most essential tools in the field of global optimization with importance ranging from conditions on optimum solutions to construction of algorithms. First and second order optimality conditions, with the gradient and Hessian of the objective function provides us with strong categorizations of local solutions to an optimization. Deterministic methods such as Conjugate Gradient, Steepest Descent, and Trust Region methods allow us to, with the gradient of the objective function, determine local (or even global) solutions with reasonable proficiency.

There are, however, some draw-backs. First of all, many real life problems are large scale problems with a lot of noise in the data, resulting in a lot of local solutions.

Classical deterministic optimization methods are derivative dependent and are non-decreasing local optimization methods. Whilst these methods allow us to solve for global optimum in some special cases[1], they are generally limited to solving for local optima depending on initial states chosen. We shall discuss this in further detail in chapter 2 of this paper.

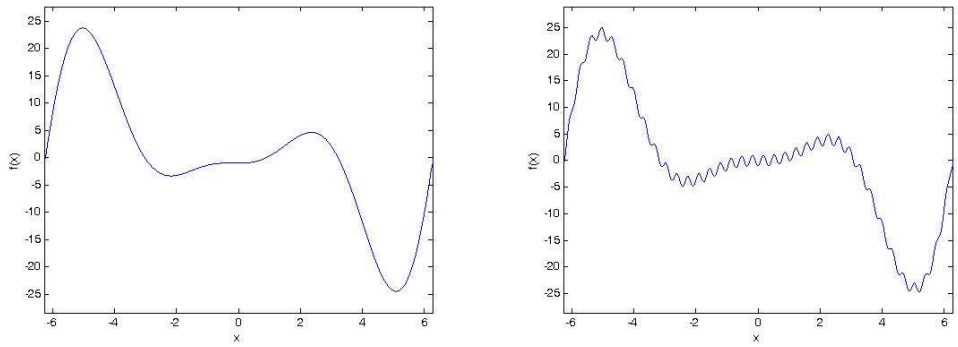


FIGURE 1.1: Example of an objective function with data noise on the right, and noise free on the left

Furthermore, in many optimization problems we simply do not have the luxury of having a reliable derivative, as the computation of derivative of the objective function is impractical in many cases.

Perhaps one of the most challenging cases is where the optimization problem is being given as a *"Black-box"* simulation model. Black-box simulation models are commonly encountered in real life situations in the form of legacy systems or sophisticated experimental simulations. In these situations, the evaluation of the objective function are sometimes expensive, and it would be unrealistic to estimate the derivative with method such as repeated finite differencing. Thus classical deterministic optimization methods will be inaccurate and inefficient.

[1] e.g. when objective function is concave and feasible region is convex

In summary, unavailability of derivatives and difficulty to evaluate objection function limits the adaptability of classical deterministic optimization methods to solve real life problems.

## 1.3 Outline of the paper

Classical deterministic methods are not effective when searching for global optimums in the following scenarios:

1. Black-box models
2. Problems with multiple local optima

We will examine how Simulated Annealing can be adapted to solve such optimization problems more effectively than classical deterministic methods. In the Appendix A, we will demonstrate a real life application of the Simulated Annealing method: a container terminal vessel loading optimization problem. Container vessels, prior to entering the port terminal, provides the port with a list of containers to be loaded onto the vessel. The objective of this problem is to determine the more efficient way of loading a single vessel. We will also discuss shortcomings of Simulated Annealing, and discuss possible ways to overcome them using techniques from Derivative Free Optimization methods.

Chapter 2 of the paper presents a survey of the Simulated Annealing method, a stochastic method inspired by the physical annealing process of metallurgy. We will explore the method's components in detail, and we will show that the method generally provides a good approximation to the global optimum.

In Chapter 3 we will investigate the inefficiencies of Simulated Annealing when dealing with non-linear continuous optimization problems. We will then discuss the Derivative Free Trust Region method, and propose new modifications to the neighborhood generating function to overcome the shortfalls of Simulated Annealing.

# Chapter 2

# Simulated Annealing

## 2.1 Overview

In this chapter, we will discuss the Simulated Annealing algorithm. We will first look at a skeletal version of a general Simulated Annealing algorithm, in particular, the heuristic nature of Simulated Annealing method, i.e. some parts of the algorithm are left undefined, which grants us the flexibility to adapt this algorithm to a great variety of problems.

We will then briefly discuss the difference between discrete and continuous optimization problems applications of Simulated Annealing, followed by a summary of convergence results from existing literatures. Proofs of the results will be omitted and can be found in the cited papers.

Lastly, we will modularize the algorithm, and explore how we could tailor each component to adapt the algorithm to various problems. We will also discuss shortcomings of Simulated Annealing, and discuss possible ways to overcome them.

## 2.2 Origin

Simulated Annealing is a meta-heuristic adaptation of the Metropolis method published by Metropolis et al. in 1953 [1]. It is an iterative stochastic method designed to address the limitations of classical deterministic methods in overcoming local optima, and to find a good approximation to the set of global solution of an optimization problem. Simulated Annealing has been a popular method in practice for its adaptability to various problems and its convergence property which mimics physical systems. The most distinguished feature of the method is the ability to escape local optima by means of hill-climbing

moves, i.e. by accepting intermediate points that might have a worse objective function value. This allows us to bypass local optimums, and eventually converge onto solutions that are at least as good as classical deterministic optimization methods.

Simulated Annealing was inspired by, and hence named after, annealing process in metallurgy. Annealing is a heat treatment technique applied to metal compound to achieve lower energy states. This is achieved by first heating the metal compound, and then letting it cool down in a controlled manner. Given an optimization problem, a cooling schedule and an initial guess, Simulated Annealing draws analogy to the physical annealing process by mimicking the heating and cooling processes. "Heating" is done by allowing a great selection of neighboring solutions of the initial guess to be accepted; "Cooling" is done by carefully reducing the choices of selection in each iteration. The nature of the annealing cooling schedule will limit our choices of solutions chosen at each iteration, and the algorithm will eventually "freeze" upon reaching the state of lowest energy (ground state): or in the optimization frame work, the global optimum.

**Example 2.1.** *The following diagram illustrates how Simulated Annealing "escapes" a local optimum:*

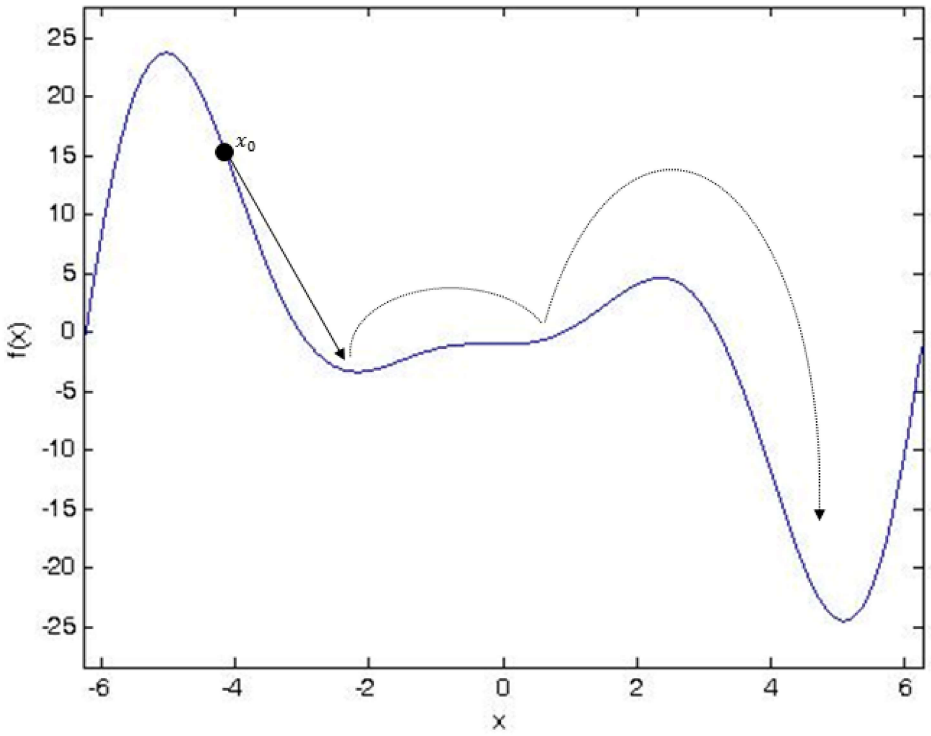


FIGURE 2.1: Simulated Annealing (dotted line) escapes local optimum (at $x \approx -2$), whilst classical down-hill method, represented by solid line, gets trapped

## 2.3 Assembly of the Algorithm

Consider a global optimization problem set up as in section 1.1:

$$\min f(x)$$
$$\text{such that } x \in \Omega$$

We establish the following general framework of Simulated Annealing according to [2–4] as follows:

**Algorithm 1** Skeletal Simulated Annealing Algorithm

**INPUT:** Initial solution $x \in \Omega$. Initial temperature $t_0$. Repetition schedule $\{M_k\}_{k\in\mathbb{N}}$. Cooling schedule $U_k^m : \{x_i\} \subseteq \Omega \to \mathbb{R}$, Acceptance function $A : \Omega\times\Omega\times\mathbb{R} \to [0,1] \subset \mathbb{R}$
**OUTPUT:** Annealing chain $\mathcal{Z} = \{x_i\}$ ▷ Also called annealing schedule

$k \leftarrow 0$; $\mathcal{Z} \leftarrow \{x\}$
**repeat** ▷ Outer loop: iterates on $k$
  Repetition counter: $m \leftarrow 0$
  **repeat** ▷ Inner loop: iterates on $m$
    Generate $x_{new} \in N(x)$ from $P_k^m(x,\cdot)$ ▷ $N(x)$ is a neighborhood of $x$
    ▷ $P_k^m(x,\cdot)$ is the candidate distribution
    Sample $p \in U[0,1]$
    **if** $p \leq A$ **then**
      $x \leftarrow x_{new}$
    **else**
      $x \leftarrow x$
    **end if**
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{x\}$
    $m \leftarrow m + 1$
  **until** $m = M_k$
  $t_{k+1} \leftarrow U(\mathcal{Z})$
  $k \leftarrow k + 1$
**until** Stopping criterion is satisfied

*Remark* 2.1. Some components of the algorithm are left undetermined, namely those that are boxed by $\boxed{\cdot}$. These undetermined components give the algorithm the flexibility to be modified to deal with a great variety of problems. We shall discuss these components in greater detail in the following section

*Remark* 2.2. Simulated Annealing is generally depicted as a *Markovian* method. That is, each step only depends on the previous iteration. Hence in the following sections we will assume, unless otherwise specified, that all functions $g$ on the annealing chain depend only on the previous iteration. In other words, $g(\mathcal{Z}) = g(x_k)$ at the $k^{th}$ iteration.

## 2.4 Survey of Convergence Criteria

In this section we will discuss convergence results from existing literatures. It is worth noting that, whilst Simulated Annealing was originally designed to tackle discrete optimization problems, we could extend it to continuous optimization problems. Our discussions in this chapter can be applied to both continuous and discrete optimization problems with the following conversion [5]:

| | Discrete problems | Continuous problems |
|---|---|---|
| $\mu$ | Uniform measure | Lebesgue measure |
| $g : \Omega \to \mathbb{R}$ | $\sum_{x\in\Omega} g(x)$ | $\int g(x)dx$ |

The components of the Simulated Annealing algorithm, namely the set of neighborhoods, cooling schedule and acceptance function, all play a significant role in determining the equilibrium distribution of the annealing schedule [6]. Therefore the convergence analysis below may vary depending on the adaptation of the algorithm. In this paper, we follow loosely the convergence analysis summarized by Henderson et al [3]:

There are two approaches to convergence results of Simulated Annealing, where the annealing chain is treated as a sequence of homogeneous Markov chain, or as an inhomogeneous Markov chain.

### 2.4.1 As a sequence of homogeneous Markov chains

The first approach assumes that, for each outer loop $k$, the number of inner loops $M_k$ is sufficiently large such that the annealing chain converges to the equilibrium distribution $\pi_k$. The temperature function $t_k$ is constant throughout each inner loop.

**Definition 2.3.** A Markov chain is ***irreducible*** if $\forall x_i, x_j \in \Omega$, $\exists n_{ij} \in \mathbb{N} \setminus 0$ such that

$$P(x_{n_{ij}} = x_j | x_0 = x_i) > 0$$

i.e. we can get from any state to any other state in a finite number of steps.

A state $\bar{x} \in \Omega$ of the Markov chain is ***aperiodic*** if $\exists n \in \mathbb{N}$ such that $\forall N \geq n$:

$$P(x_N = \bar{x} | x_0 = \bar{x}) > 0$$

i.e. state $\bar{x}$ returns to itself at irregular times. An irreducible Markov chain $\mathcal{Z}$ is aperiodic if there exists one aperiodic state $\bar{X} \in \mathcal{Z}$.

A finite Markov chain is ***regular*** if and only if it is both ***irreducible*** and ***aperiodic***.

A Markov chain is ***reversible*** if for all states $x, y \in \Omega$ and all iteration $k$:

$$\pi_k(x)P(x_{n+1} = y|x_n = x) = \pi_k(y)P(x_{n+1} = x|x_n = y)$$

The following theorem by Çinlar [3, 7] asserts the existence uniqueness of the equilibrium distribution:

**Theorem 2.4** (Çinlar, 1974)**.** *For states $x, y \in \Omega$, let $P^{(m,k)}(x, y)$ denote the transitional probability matrix from $x$ to $y$ in $k$ inner loops and $m$ outer loops. If the Markov chain $\mathcal{Z}$ defined by $P^{(m,k)}$ is regular, then the following limit exists for all $x, y \in \Omega$, for all $k$:*

$$\pi_k(y) := \lim_{m\to\infty} P^{(m,k)}(x, y)$$

*Moreover $\pi_k(y)$ the unique strictly positive solution of:*

$$\pi_k(y) = \sum_{x\in\Omega} \pi_k(x)P_k(x, y) \tag{$\star_1$}$$

$$\sum_{x\in\Omega} \pi_k(x) = 1 \tag{$\star_2$}$$

*Remark* 2.5. Depending on the adaptation of Simulated Annealing, proofs comes in different flavor. However, when the the annealing chain is treated as a sequence of homogeneous Markov chains, regularity and reversibility of the annealing chain is essential to guarantee the uniqueness of the stationary distribution $\pi_k$. In fact, as Henderson et al [3] remarked, reversibility of the annealing chain is the sufficient condition of all such proofs of convergence.

If the reversibility condition is not satisfied, the explicit form of the stationary distribution $\pi_k$ would be very difficult to compute, as this typically involves solving large linear systems $(\star_1)$ and $(\star_2)$ [8].

### 2.4.2 As a inhomogeneous Markov chain

The second approach treats the annealing chain as a single inhomogeneous Markov chain. This approach no longer requires an arbitrarily large inner loop to assert convergence to the stationary distribution, and instead considers the limiting behavior of the annealing chain on the outer loop.

The proof of convergence by Mitra et al [9] requires the annealing chain to satisfy the weak and strong ergodicity, as well as existence of eigenvectors $\pi_k$ in the form given by $(\star_1)$ and $(\star_2)$ in theorem 2.4. The proof also requires $\pi_k$ to converge to the equilibrium

distribution $\pi^{opt}$, where $\pi^{opt}$ is the probability distribution where only global optima has probability greater than 0.

The theorem by Hajek [10] below, on the other hand, imposed a lower bound on the probability to escape local optima (but not global optima). Hajek furthermore showed that the logarithmic cooling schedule is a necessary and sufficient condition for convergence [11].

**Definition 2.6.** Given $x \in \Omega$, a neighborhood $N(x)$ of $x$, and iteration counters $m, k \in \mathbb{N}$ (the $k^{th}$ outer loop and the $m^{th}$ inner loop, the ***candidate distribution*** (or the ***generation probability distribution***) is described by: $P_k^m(x, y) = P_k^m(x_\zeta = y | x_i = x)$, the probability distribution function over $N(x)$ for generating a point $y \in N(x)$ from $x$.

**Theorem 2.7** (Hajek, 1988)**.**

$$d^* := \min_{x \in \Omega} \{\max \{f(\mathcal{P}(x, \Omega^*)\} - f(x)\}$$

*where $\Omega^*$ is the set of global optimums, and $\max \{f(\mathcal{P}(x, \Omega^*)\}$ is the largest function value along the path from $x$ to $\Omega^*$. In other words, $d^*$ largest depth of local optimum that is not a global optimum [12].*

*Then the Simulated Annealing algorithm converges if and only if*

1. $\lim_{k \to \infty} U_k^m = 0$
2. $\sum_k \exp\left\{\frac{-d^*}{U_k^m}\right\} = \infty$

*Remark* 2.8. In particular, annealing chain with the following cooling schedule converges;

$$U_k^m = \frac{c}{\log(k + k_0)}$$

where $c \geq d^*$, and $k_0 \geq 0$ is a positive constant. This is known as the ***logarithmic cooling schedule***. and it provides us with a necessary condition of convergence. We shall discuss cooling schedules in greater detail in section 2.5.2.

## 2.5 Component Analysis

From Algorithm 1, we have identified the components of Simulated Annealing that we must specify when dealing with a given optimization problem in practice. In the following sections, we shall analyze each of these components and discuss how we could temper these components to adapt the method to various problems.

### 2.5.1 Acceptance Function

The most distinguished feature of Simulated Annealing lies in its ability to perform hill climbing moves to escape local optima. That is, the ability to accept intermediate points in the feasible region that worsen the objective function value.

In this section we will see how the algorithm decides whether or not to accept new candidate points by defining a reasonable acceptance rule.

Franz et al. [6, 13] provided us with a general framework for a general acceptance rule for the acceptance function $A$ should satisfy (for the $k^{th}$ iteration):

$$A : \Omega \times \Omega \times \mathbb{R} \to [0,1] \subset \mathbb{R}$$
$$\{x_k, x_{new}, t_k\} \mapsto q \in [0,1]$$

1. $A$ is a function on $\Delta f := f(x_{new}) - f(x_k)$, and the temperature $t_k$
2. At $t_k = \infty$, all moves will be accepted, i.e. $A(\Delta f, \infty) = 1$.
3. For a fixed $t_k < \infty$:
   (a) Downhill moves are always accepted, i.e. $\Delta f \leq 0 \Rightarrow A = 1$
   (b) Uphill moves can be accepted with acceptance probability monotone decreasing with respect to $\Delta f$.
   (c) Drastic uphill moves are rarely accepted. i.e. $\Delta f \to \infty \Rightarrow A \to 0$.

In other words:

1. $A$ is monotone increasing with respect to temperature $T \subset \mathbb{R}$
2. $A$ is monotone decreasing with respect to $\Delta f := f(x_{new}) - f(x_k)$

We will look at a few examples of acceptance rules in existing literature.

In lieu of the origin of the method, most literature uses the ***Metropolis criterion*** as the acceptance rule:

$$P_M := A(x_k, x_{new}, t_k) = \min\left\{1, \exp\left(\frac{-\Delta f}{t_k}\right)\right\} \tag{2.1}$$

Equivalently, we would have the following in the Simulated Annealing algorithm:

```
if f(x_new) ≤ f(x_k) then
    x_{k+1} ← x_new
else
Generate p ∈ U[0, 1]
    if exp(−Δf / t_k) < p then
        x_{k+1} ← x_new
    else
        x_{k+1} ← x_k
    end if
end if
```

*Remark* 2.9. The total probability of accepting uphill moves generated by the Metropolis criterion is exactly $t_k$:

$$\int_0^\infty P_M(\Delta f)d(\Delta f) = t_k$$

In other words, the Metropolis criterion, whilst always accepting downhill moves, has a chance of accepting uphill moves as well. This provides us with the flexibility to avoid being trapped in local optima. Moreover, as $t_k \to 0$, $\exp\left(\frac{-\Delta f}{t_k}\right) \to 0$, and therefore we will have less and less uphill moves as the system "cools down", eventually converging onto a global optimum. These observations agree with the general properties stated in the beginning of the section.

At a constant temperature function $U(t_k)$, the equilibrium distribution $\pi_k$ of the annealing chain is given by the Boltzmann distribution [6].

Depending on the nature of adaptation, there are a variety of acceptance rules. One of the variations of the Metropolis function is the ***Barker criterion***:

$$P_B := A(x_k, x_{new}, t_k) = \frac{1}{1 + \exp\left(\frac{\Delta f}{t_k}\right)}$$

The annealing chain under Barker criterion has the same stationary distribution as Metropolis criterion at a constant temperature function. However, authors of [14, 15] are able to derive a faster algorithm while varying the neighborhood candidate distribution with a cooling schedule (which we shall discuss in the next section). This method is thus called fast annealing, and the annealing chain satisfies FermiDirac distribution [6], which leads to a faster convergence.

In lieu of the enhancements achieved by the fast annealing algorithm, it would be natural to ask whether an optimal acceptance function exists. In particular. Franz et al. [13] proved that, if $\Omega$ is finite, and if the objective function $f$ depends linearly on the final

probability (or the objective function value) of the global optimum, then the following holds:

$$P_F := A(x_k, x_{new}, t_k) = \begin{cases} 1 & \text{if } \Delta f \leq 0 \\ \frac{1}{\mu(N(x))} & \text{if } \Delta f > 0 \text{ and } \frac{1-q}{2-q}\frac{\Delta f}{t_k} \leq 1 \\ 0 & \text{if } \Delta f > 0 \text{ and } \frac{1-q}{2-q}\frac{\Delta f}{t_k} > 1 \end{cases}$$

The annealing chain performs optimally in the limiting case when $q = -\infty$, where $P_F$ becomes the ***Threshold acceptance criterion*** [16, 17]:

$$P_{Threshold} := A(x_k, x_{new}, t_k) = \begin{cases} 1 & \text{if } \Delta f \leq t_k \\ 0 & \text{otherwise} \end{cases}$$

Just like Metropolis criterion, the total probability of accepting uphill moves by threshold acceptance also equals to $t_k$, i.e. it also satisfies the conditions in Remark 2.9.

*Remark* 2.10. It is worth noting that, despite the optimality of threshold acceptance, other acceptance rules may be preferred depending on the adaptation and implementation of the algorithm.

### 2.5.2 Temperature schedule

In this section we will look at the temperature schedule of an annealing chain, which can generally be described by the following components:

1. Initial temperature $t_0$
2. Cooling schedule $U : \{x_i\} = \mathcal{Z} \subseteq \Omega \to \mathbb{R}$
3. Repetition schedule $\{M_k\}_{k \in \mathbb{N}}$
4. Stopping criterion

*Remark* 2.11. In practice, we would naturally want the algorithm to terminate in finite time steps, this implies that the generated annealing chain $\mathcal{Z}$ will be at most finite.

In the following discussion we may assume, without lost of generality, that the state space $\Omega$ is finite (though it could be arbitrarily large).

### 2.5.3 Initial temperature and Stopping criterion

The initial temperature and the stopping criterion is generally determined by the physical nature of the problem.

The initial temperature $t_0$ is usually defined to be sufficiently large such that almost all moves from the initial guess will be accepted.

The stopping criterion on the other hand, comes with more variety and requires different sets of computation for different cooling schedules. The general rule of thumb based on existing literatures is to terminate when no significant progress can be made after a certain number of iterations [4].

### 2.5.4 Repetition schedule

The implementation of a repetition schedule is optional, and is generally set up according to the adaptation of the algorithm. It is generally employed for convergence purposes as described in section 2.4.1.

### 2.5.5 Cooling schedule

A good cooling schedule is crucial to the performance of the Simulated Annealing algorithm. Fast cooling schedules enable fast convergence but depending on the nature of the problem, may restrict the range of feasible region explored. Slower cooling schedules, on the other hand, allows more feasible region to be explored, and hence a better chance of obtaining a good approximation to the global solution.

Therefore an appropriate rate of cooling determines solution quality and algorithm efficiency. In practice, the rate of cooling depends largely on the problem specifications and the user's preference.

Ideally we would prefer cooling schedules that assert convergence of the algorithm to the set of global optima. However, a study conducted by Cohn and Fielding [12] suggests that convergent cooling schedules are too slow, while repeated independent executions of the algorithm with non-convergent cooling schedules still provide reasonable results.

Cooling schedules can be categorized into two groups: *static schedule* and *adaptive schedules*. Static schedules are set prior to the execution of the algorithm. Adaptive schedules, on the other hand, adjusts the rate of cooling during the execution or between executions of the algorithm according to information obtained.

#### 2.5.5.1 Static Cooling Schedule

Static schedules are defined completely prior to the execution of the algorithm, and are typically dependent on a control parameter that allows users to adjust the rate of cooling. We present below two of the most popular cooling schedules:

One of the most popular choice of cooling schedules is given by the ***exponential schedule***:

$$U_k^m = t_0 \cdot \alpha^k$$

where $\alpha \lesssim 1$ is called the cooling factor.

And from Hajek's theorem (Theorem 2.7), we define the ***logarithmic cooling schedule*** as follows:

$$U_k^m = \frac{c}{\log(k + k_0)}$$

where $c \geq d^*$, and $k_0 \geq 0$ is a positive constant. This not only provides us with a necessary condition of convergence, but a study of Cohn and Fielding [12] also suggests that critical points of the limiting behavior of the annealing chain occurs when the cooling schedule is close to the logarithmic schedule.

Intuitively the quality of the cooling schedule improves as we utilize more information of the objective function. Hence we will look into adaptive cooling schedules, which aim to optimize the rate of cooling by exploiting the information of the annealing chain.

#### 2.5.5.2 Adaptive Cooling Schedule

Adaptive schedules can be implemented either during the execution or between multiple runs of the algorithm, where the latter is generally more popular due to ease of parallelization. As proposed by [6], we may adjust the schedule according to the rate of convergence, objective function value, or both. There are various forms of adaptive cooling schedules depending on the implementation of the algorithm, and we will discuss an example defined by Bohachevsky et al [18] as follows:

$$U_k^m = \beta \left[ f(x_k) - \hat{f} \right]^g$$

where $\beta, g > 0$ are constants, and $\hat{f}$ is an estimate of the optimal objective function value. The estimate $\hat{f}$ is adjusted according to the objective function value during the

execution of the algorithm subject to the rules: (for a minimization problem) $\hat{f} \leq f(x_k)$ for all points $x_k$ visited, and if $f(x_k) - \hat{f} \gg 0$ then $\hat{f}$ can be increased.

This allows up-hill moves to occur more frequently when the current objective function value is far from optimal value. And on the other hand, when the current objective function value is close to the estimated optimal objective function value $\hat{f}$, fewer up-hill moves will be accepted. This allows us to further avoid being trapped in local optima.

Though the cooling schedules described above are popular for their simplicity, they both are non-increasing functions of $k$, which is not ideal for most optimization problems. In fact, the experiments conducted by Strenski and Kirkpatrick [19] suggests that the ideal cooling schedule are not monotone decreasing.

Lastly, it is worth noting that the choice of cooling schedule depends largely on the empirical behaviors of the optimization problem. Although the general intuition is that the more information we utilize, the better the performance of cooling schedule, inferior schedules may yield a better performance subject to the nature of the optimization problem.

### 2.5.6 Neighborhood and Candidate Distribution

Recall from Algorithm 1: each iteration a new point $x_{new}$ is generated from $N(x)$, a neighborhood of the current point $x$, with candidate distribution $P_k^m(x, \cdot)$ (from definition 2.6). The algorithm then determines whether to accept or reject the new point according to the acceptance function, which we have discussed in section 2.5.1.

The neighborhood and candidate distribution will be defined naturally by the following function:

$$\begin{aligned} \mathcal{N} : \Omega &\to \mathcal{D} \subseteq \mathcal{P}(\Omega) \\ x &\mapsto \{N(x)\} = \mathcal{N}(x) \end{aligned}$$

where $\mathcal{P}(\Omega)$ is the power set of $\Omega$. Given a point $x \in \Omega$, $N(x)$ is chosen from the subset $\mathcal{N}(x)$ of $\mathcal{P}(\Omega)$, and the candidate distribution function $P_k^m(x, \cdot)$ is a probability distribution over $N(x)$.

This suggests that the neighborhood function $\mathcal{N}(x)$ is therefore determined by the topology of the feasible region and the objective function. In other words, the neighborhood function is determined by the geometry and physical nature of the optimization problem. Hence, the choice of neighborhood functions are usually quite restrictive.

The choice of candidate distribution, however, provides us with more flexibility.

For the rest of the section we will focus on generating points from a neighborhood. We further partition our discussion into two parts. In the first part of the discussion, we will assume we already have a given a neighborhood $N(x) \in \mathcal{N}(x)$ of a given point $x$. We will look at a couple choices of candidate distribution $P_k^m(x, \cdot)$ to generate new points from $N(x)$. In the second part of the discussion, we will look at ways to choose a "good" neighborhood from the set $\mathcal{N}(x)$.

#### 2.5.6.1 Generation of points

Suppose we are given a point $x \in \Omega$, and a neighborhood $N(x)$ of $x$. We will present and discuss possible choices of candidate distribution according to Dekkers and Aarts [20], Locatelli [4], and Henderson et al [3]:

As discussed insection 2.4, the Markov chain generated by the Simulated Annealing algorithm (the annealing chain) must be both *regular* and *reversible*.

Therefore it is natural to choose an isotropic distribution over the neighborhood $N(x)$. In particular, one of the most natural choices of such candidate distribution would be the uniform distribution over $N(x)$:

$$P_k^m(x, y) = \frac{1}{\mu(N(x))}, \quad \forall y \in N(x), \forall k, \forall m \tag{2.2}$$

where $\mu$ is either the Lebesgue measure on or the uniform measure on $N(x)$ for continuous or discrete problems respectively, as we have discussed earlier on in section 2.4.

This candidate distribution function allows us to examine the entire neighborhood in an unbiased fashion. Moreover, the annealing chain generated by this probability distribution function is reversible and regular, and therefore ensures convergences of the Simulated Annealing algorithm.

However, this candidate distribution does not consider any information of the neighborhood $N(x)$, and assumes that the objective function has the same behavior for the entire $N(x)$, which is often not the case.

Dekkers and Aarts [20] thus proposes an alternative:

$$P_k^m(x, y) = \begin{cases} LS(x) & \text{if } w > t \\ \frac{1}{\mu(N(x))} & \text{otherwise} \end{cases}, \quad \forall y \in N(x), \forall k, \forall m \tag{2.3}$$

where $t \in [0, 1)$ is fixed, and $w \in U[0, 1)$. $LS(x)$ is an arbitrary local descent directional search method that generates a point from $x$.

Or equivalently we would have the following algorithmic form:

Fix $t \in [0, 1)$
Generate $w \in U[0, 1]$

1: **if** $w > t$ **then**
2: $\quad y \leftarrow LS(x)$
3: **else**
4: $\quad P_k^m(x, y) = \frac{1}{\mu(N(x))}, \ \forall y \in N(x), \forall k, \forall m$
5: **end if**

It was shown, by Dekkers and Aarts [20], that despite the fact that the annealing chain generated by this candidate distribution is not reversible (in (2.3): $P_k^m(x, y) \neq P_k^m(y, x)$), it still converges to a good approximation to the global optimum.

However, classical local search methods rely heavily on information of the objective function. In particular for continuous functions, we require at least the derivative of the objective function. Therefore there is no easy way to apply classical local search methods to a Black-box model.

In Chapter 3, we will further address this issue with A Derivative Free local search method known as Derivative Free Trust Region method.

For the rest of the paper we will stick to the uniform distribution as in equation (2.2) for the candidate distribution.

#### 2.5.6.2 Choice of Neighborhood

Given a point $x \in \Omega$, $|\mathcal{N}(x)| \geq 2$, since $\{x\} \subsetneq \Omega \in \mathcal{N}(x)$. This provides us with some flexibility when choosing a neighborhood $N(x)$.

The choice of neighborhood comes in two options, neighborhoods can be chosen by size or by the topology, which depends on the nature of the optimization problem. Choosing a "good" neighborhood is an essential to designing an efficient Simulated Annealing algorithm [21].

Moreover, to ensure the quality of approximation of and the efficiency of finding a global optimum, the size of the neighborhood plays an important roll in limiting the choices of points during the cooling process. The rate of decrease in size of neighborhood per iteration therefore affects the rate of convergence of Simulated Annealing.

The most common approach is to scale the size of the neighborhood according to the temperature parameter $t_k$. i.e. the size of neighborhood is a function that increases

monotone with respect to the temperature parameter $t$. We shall observe a similar behavior exhibited by the Trust Region radius in chapter 3.

The specific size function, however, would vary from problem to problem, and should be scaled according to the desired rate of convergence.

On the other hand, we have more flexibility in terms of the topological structure we can impose on the neighborhoods.

Discrete optimization problems are typically motivated by physical problems, and the neighborhood function is generally completely determined by the nature of the problem tackled. Continuous problems, on the other hand, provide us with more interesting variety and flexibility in terms of the topology we can impose on the neighborhood function.

For the rest of the discussion we will, without lost of generality, assume $\Omega = \mathbb{R}^n$ (implying $\mathcal{N}(x) = \mathcal{P}(\mathbb{R}^n)\ \forall x$), in other words we will be solving the following unconstrained continuous optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x)$$

where $f$ is locally differentiable, i.e. the gradient $\nabla f(x)$ exists for all $x$.

For a Black-box model, without any prior knowledge of the objective function whatsoever, the general approach would be to let $N(x)$ be of unit size in all directions (i.e. a unit $(n-1)$ dimensional sphere for continuous problems). This allows us to explore the feasible region $\Omega$ in an unbiased manner as per the discussion for a uniform candidate distribution in (2.2). As was pointed out in previous discussion by Locatelli [4], Vanderbilt and Louie [22], objective functions generally do not exhibit the same behavior in all directions, hence a unit sphere approach would fail to encapsulate the topological information of the system.

We illustrate this issue with following diagrams inspired by Locatelli [4]:

Suppose the entire contour $\nabla f$ of $f$ (level set of $f$) is given (Fig 2.2 and Fig 2.3), with starting point $x$.

In Fig 2.2, the neighborhood $N(x)$ are chosen to be $(n-1)$ dimensional sphere of two different sizes ($N_1(x)$ and $N_2(x)$).

Notice that $f$ changes slowly in $x^1$, the only descent direction of $f$ from $x$. If we generate a new point using the uniform candidate distribution function on $N_1(x)$, chances of obtaining a better solution is quite low. Hence the new point will likely be rejected.

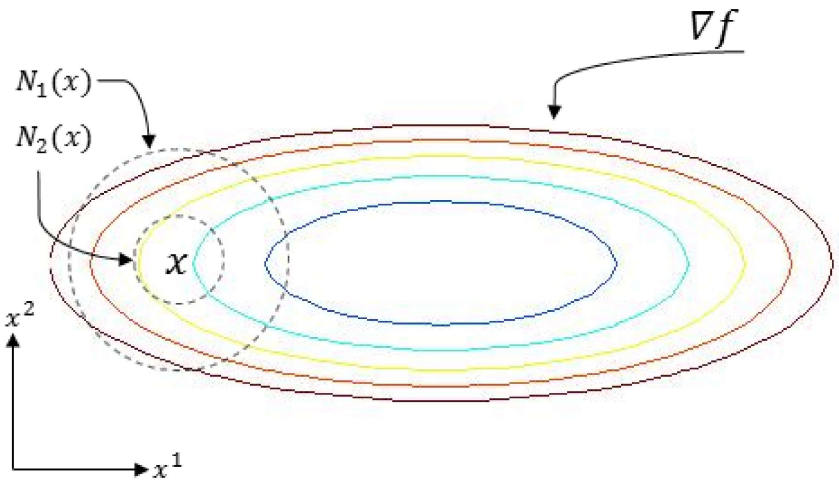


FIGURE 2.2: Due to the incoherence between neighborhoods $N_1(x)$ or $N_2(x)$ and the contours $\nabla f$, new points sampled will likely be rejected.

Furthermore, this problem persists even if the step size decreases, and new points are sample from a smaller neighborhood illustrated by $N_2(x)$ in Fig 2.2.

Therefore the choice of neighborhood must take into consideration the topological information of the objective function, and the feasible region should be searched *anisotropicly*, mimicking the geometry of the objective function.

In Fig 2.3, the neighborhood $N_3(x)$ is chosen such that the support of $N_3(x)$ approximates the shape of the contours $\nabla f$.

This choice of neighborhood takes into consideration the topology of the objective function, and allows us to sample points skewed towards the direction of steepest descent. This allows us to take larger steps in directions with slower change ($x^1$), and smaller steps in directions with faster change ($x^2$), which is in essence a uniform search bias towards local optima that satisfies first order optimality conditions

To find a neighborhood such as $N_3(x)$, Vanderbilt and Louie [22] proposed the following approach that is similar to a quasi-Newton method:

Assume Hessian of the global optimum $x^*$ is known and positive definite:

$$H^* := H(x^*)$$

The new point $x_{new}$ will be generated as follows:

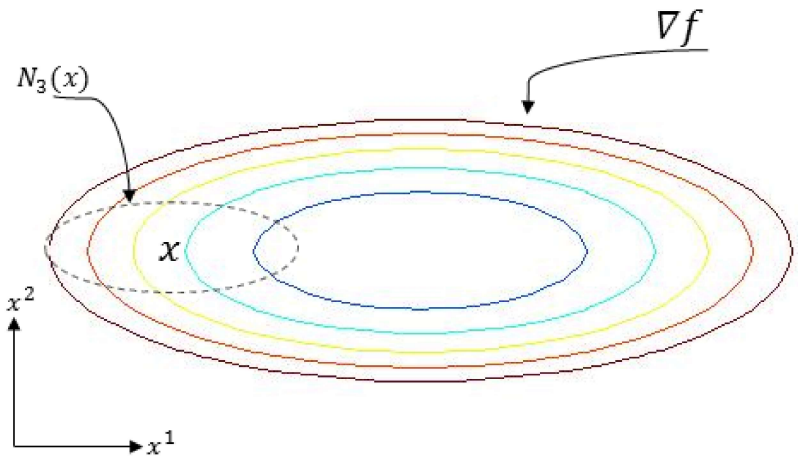


FIGURE 2.3: $N_3(x)$ is chosen to approximate the contours $\nabla f$, and has a better chance of obtaining more desirable points along the $x^1$ axis

**INPUT:** Starting point $x \in \Omega$, Hessian of global optimum $H^*$.

**OUTPUT:** A new point $x_{new}$.

1: Compute $S := (H^*)^{-1}$

2: Compute $S = QQ^\top$ ▷ $Q$ is the Cholesky factor of $S$

3: Generate $r \in U[\alpha, \beta]^n$ ▷ $\alpha, \beta \in \mathbb{R}^n$, $[\alpha, \beta]^n$ is a hypercube in $\mathbb{R}^n$

4: $x_{new} \leftarrow x + Q \cdot r$ ▷ Skewing the hypercube as in Fig 2.3

In the black-box model, even though the gradient $\nabla f$ exists, it is not available to us, let alone obtaining any information of $H^*$.

Vanderbilt and Louie [22] hence proposed the following method to approximate $S$, and the geometry of $f$ by utilizing points generated from the algorithm:

Suppose each outer loop of Algorithm 1 has size $M$ (where $M$ is sufficiently large), in other words $M_k = M$ for all $k$. At the end of the $\ell^{th}$ outer loop, we obtain a segment of the annealing chain $\left\{x^{\ell M+1}, \ldots, x^{\ell M+M}\right\} \mathcal{Z}_\ell \subset \mathcal{Z}$, thus we compute $A^\ell$ and $\bar{S}^\ell$, the first and second moment of $\mathcal{Z}_\ell$ respectively as follows:

$$A_i^\ell = \frac{1}{M} \sum_{k=1}^{M} x_i^{\ell M+k}$$

$$\bar{S}_{ij}^\ell = \frac{1}{M} \sum_{k=1}^{M} \left[x_i^{\ell M+k} - A_i^\ell\right] \cdot \left[x_j^{\ell M+k} - A_j^\ell\right]$$

Thus at the $(\ell+1)^{th}$ outer loop, we computer the matrix $S := S^{(\ell+1)}$ by:

$$S^{(\ell+1)} = \frac{\chi_S}{\beta M}\bar{S}^{\ell}$$

where $\chi_S > 1$ is called the growth factor, and $\beta > 0$ is based on the geometric average of the second moment.

By computing the Cholesky factor $Q$ of $S \approx H^{-1}$, this method draws analogy to quasi-Newton methods. As the search direction $\Delta x = Q \cdot r$ draws strong correlations to the quasi-Newton direction $\Delta x_{\mathrm{QN}} = H^{-1} \cdot -\nabla f$. However, quasi-Newton methods do not guarantee second order optimality conditions.

## 2.6 Summary

Simulated Annealing is a highly adaptive optimization method with wide practical application. As was pointed out earlier, typical Simulated Annealing is based on pure random search (candidate points are sampled uniformly over isotropic neighborhood systems) when no prior knowledge of the optimization problem is available. Hence the major drawback of Simulated Annealing is the slow theoretical convergence rate primarily due to its insensitivity towards topological information. In Appendix A, we will demonstrate a real life application of Simulated Annealing method to a container port terminal optimization problem.

Global optimization problems with strong non-linearity will therefore have to be tackled with techniques from Derivative Free Optimization, which we will discuss in the following chapter.

# Chapter 3

# Derivative Free Optimization

## 3.1 Introduction

The major draw back of the Simulated Annealing method is that it overlooks topological information when choosing the neighborhood function. Simulated Annealing method generally cannot determine a "good" choice of neighborhood function during execution of the algorithm, which leads to the method's theoretical inefficiency when solving optimization problems with strong non-linearity. In this chapter we will discuss various methods to improve this particular aspect of Simulated Annealing by techniques of Derivative Free Optimization. We will then present a new method which generates search direction utilizing topological information of the objection function $f$.

In this chapter, we will consider the unconstrained optimization problem of the following form:

$$\min_{x \in \mathbb{R}^n} f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear function that is "sufficiently smooth". i.e. $\nabla^i f(x)$ exists for all $x \in \mathbb{R}$ and is continuous up to some $i \geq 1$, and $\nabla^{i+1} f(x)$ is Lipschitz continuous, even though they cannot be computed or approximated directly.

Derivative Free optimization can be roughly categorized into several classes [23, 24]: directional simplicial search methods such as Nelder Mead [25] and generalized pattern search [26]; line search methods such as Powell's method [27]; and sampling methods such as implicit filtering and trust region methods based on polynomial interpolation.

We will discuss ways to utilize the search direction generated by these different classes of Derivative Free Optimization methods to increase the efficiency of Simulated Annealing.

*Remark* 3.1. The general approach is to utilize the search direction $s_k$ generated by Derivative Free Optimization to Simulated Annealing method to generate a ellipsoidal neighborhood similar to that described in Fig 2.2 and 2.3:

1. In each iteration in Simulated Annealing, use Derivative Free optimization method to generate a search direction $s_k$.
2. Skew the unit $(n-1)$ dimensional sphere by the direction $\pm s_k$, and search for a new candidate point $x_{new}$ uniformly in the new neighborhood.

In this chapter we will discuss how to obtain a particular search direction by Derivative Free Trust Region method. However, it is worth noting that any method that generates a reasonable search direction can be applied in a similar manner.

## 3.2 Derivative Free Trust Region method

Derivative Free Trust Region methods were introduced by Powell [28] and Winfield [29]; they are an approximation based optimization method that utilized trust region technique, and approximation models of the objective function.

Derivative Free Trust Region methods comes in different flavors, depending on the approximation method used. We will focus on the *polynomial interpolation* based approximation methods described by Conn, Scheinberg and Vicente [24].

To construct the polynomial interpolation based Derivative Free Trust Region method, we first look at how to determine a "good" interpolation set $Y$. This is done by examining a property called $\Lambda$-poisedness of the interpolation set $Y$ on a closed ball $B \supset Y$.

We will then see, in the second degree case, that the error of the approximation of the $i^{th}$ derivative is bounded by $\Lambda$ and $\Delta(Y)$, the diameter of $Y$ and the trust region radius.

Then we will discuss two *model improvement* algorithms to explicitly construct, maintain, and improve the poisedness of a given set of interpolation points $Y$ which may not be poised.

Finally we will discuss the main algorithm of the quadratic interpolation Derivative Free Trust Region method, and from the sufficient conditions of global convergence, we will derive a reasonable search direction.

### 3.2.1 Interpolation model and Poisedness

Before we establish the main algorithm of an interpolation based Derivative Free Trust Region method, we must first construct a "good" interpolation model for the objective function $f$.

Consider a sample set of interpolation points $Y = \{y^0, \dots, y^p\} \subset \mathcal{B} \subseteq \Omega$, where $\Omega$ denotes the feasible region.

Let $\mathcal{P}_n^d$ denote the space of polynomials of degree less than or equal to $d$ in $\mathbb{R}^n$. Suppose $m(x) \in \mathcal{P}_n^d$ interpolate $f$ at $Y$, then it satisfies the interpolation conditions:

$$m(y^i) = f(y^i), \forall y^i \in Y \tag{3.1}$$

Let $\phi = \{\phi_0(x), \dots, \phi_q(x)\} \subseteq \mathcal{P}_n^d$ be a basis of $q+1 = \binom{n+d}{n}$ polynomials in $\mathcal{P}_n^d$. We can rewrite $m(x)$ in the following form:

$$m(x) = \sum_{j=0}^{q} \alpha_j \cdot \phi_j(x)$$

where $\alpha_j$ are constants. Hence (3.1) can be expressed in the following form:

$$m(y^i) = \sum_{j=0}^{q} \alpha_j \cdot \phi_j(y^i) = f(y^i), \forall y^i \in Y$$

Or equivalently we have the following matrix form:

$$\begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_q(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_q(y^p) \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_q \end{bmatrix} = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^p) \end{bmatrix} \tag{3.2}$$

For the rest of the chapter, the Vandermonde matrix in the equation above will be denoted by:

$$M(\phi, Y) := \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_q(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_q(y^p) \end{bmatrix}$$

Moreover, for simplicity we further abbreviate $M := M(\bar{\phi}, Y)$ if $\phi = \bar{\phi}$, where $\bar{\phi}$ is the natural basis in $\mathcal{P}_n^d$. The natural basis of monomials over $\mathbb{R}^n$ is given by (for $x =$

$(x_1, \ldots, x_n) \in \mathbb{R}^n$) [25]:

$$\bar{\phi} := \{\bar{\phi}_i\}_{i=0}^{p}$$
$$\text{where } \bar{\phi}_i := \frac{x^{\alpha^i}}{(\alpha^i)!},\ |\alpha^i| \leq d$$

In other words:

$$\bar{\phi} := \left\{1, x_1, x_2, \ldots, x_n, \frac{x_1^2}{2}, x_1x_2, \ldots, \frac{x_{n-1}^{d-1}x_n}{(d-1)!}, \frac{x_n^d}{d!}\right\} \tag{3.3}$$

To determine the interpolation polynomial $m(x)$, we compute the coefficients $\alpha_j$ by solving the linear system (3.2). In particular, the linear system has a unique solution when the matrix $M(\phi, Y)$ is non-singular. Hence we have the following definition from [24, 25]:

**Definition 3.2.** A set of interpolation points $Y = \{y^0, \ldots, y^p\}$ is called ***poised*** (or ***d-unisolvent*** in [24, 30]) for polynomial interpolation in $\mathbb{R}^n$ if $M(\phi, Y)$ is non-singular for some basis $\phi$ of $\mathcal{P}_n^d$.

*Remark* 3.3. The notion of poisedness can be viewed as a non-linear version of affine independence. In particular if $d = 1$, $Y$ is poised in $\mathcal{P}_n^1$ if and only if $Y$ is affine independent.

*Remark* 3.4. It is clear that if $Y$ is poised, then we must have (for $M(\phi, Y)$ non-singular):

$$|Y| = p + 1 = \binom{n+d}{d}$$

where $\binom{n+d}{d}$ is the dimension of $\mathcal{P}_n^d$. Moreover, if $M(\phi, Y)$ is non-singular for some basis $\phi$, then it is non-singular for any basis $\mathcal{P}_n^d$, and the notion of poisedness is independent of $\phi$.

Hence the following result arises naturally [25]:

**Lemma 3.5.** *Given a function $f : \mathbb{R}^n \to \mathbb{R}$, and a poised set $Y$, there exists a unique interpolation polynomial $m(x) \in \mathcal{P}_n^d$.*

The quality of the interpolation polynomial $m(x)$ thus depends on the choices of the set of interpolation points $Y$. Hence it is natural to define a notion of "well" poisedness of $Y$. In particularly we define such notion by the basis of Lagrange polynomials as follows [24]:

**Definition 3.6.** Given a set of interpolation points $Y = \{y^0, \ldots, y^p\}$, a basis $\{\ell_j(x)\}_{j=0}^p$ of $\mathcal{P}_n^d$ is called a ***basis of Lagrange polynomials*** if:

$$\ell_j(y^i) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

**Lemma 3.7.** *If $Y$ is poised, then there exists a unique basis of Lagrange polynomials $\{\ell_j(x)\}_{j=0}^p$ of $\mathcal{P}_n^d$ specified by $Y$ as in definition 3.6.*

*Remark* 3.8. Alternatively, we can define the basis of Lagrange polynomials as follows [24]: Given a set of poised interpolation points $Y = \{y^0, \ldots, y^p\}$, let $\phi = \{\phi_0(x), \ldots, \phi_p(x)\}$ be a basis of $\mathcal{P}_n^d$. By abuse of notation, we can let $\phi(x) = [\phi_0(x), \ldots, \phi_p(x)]^\top$ be a vector in $\mathbb{R}^{(p+1)}$.

Since $Y$ is poised, $M(\phi, Y)$ is non-singular, and therefore $\{\phi(y^i)\}$ spans $\phi(\text{Conv}(Y))$, where $\text{Conv}(Y)$ is the convex hull of $Y$. Hence for any $x$ in the convex hull of $Y$, we can express $\phi(x)$ uniquely by:

$$\phi(x) = \sum_{i=0}^{p} \lambda_i(x)\phi(y^i)$$

or equivalent we have the following matrix form:

$$\phi(x) = M(\phi, Y)^\top \lambda(x)$$

where $\lambda(x) = [\lambda_0(x), \ldots, \lambda_p(x)]^\top$ is a vector in of polynomials of degree at most $d$, and $\{\lambda_i(x)\}_{i=0}^p$ is the basis of Lagrange polynomials defined as in definition 3.6.

The basis of Lagrange polynomials provides us with an important measurement of poisedness of the interpolation set $Y$, and the quality of the interpolation polynomial $m(x)$.

In particular, Ciarlet and Raviart showed that (in Theorem 1, [30]): given a function $f$, a poised interpolation set $Y = \{y^0, \ldots, y^p\}$, and the interpolation polynomial $m(x)$ of f; for any $x$ in $\text{Conv}(Y)$, the convex hull of $Y$:

$$\|\mathcal{D}^k m(x) - \mathcal{D}^k f(x)\| \leq \frac{1}{(d+1)!} G \sum_{i=0}^{p} \|y^i - x\|^{d+1} |\mathcal{D}^k \ell_i(x)| \tag{3.4}$$

where $\mathcal{D}^k g(x)$ is the $k^{th}$ derivative of a function $g(x)$, and $G$ is the upper bound of $\mathcal{D}^{d+1} f(x)$.

Suppose, without lost of generality, that $Y$ is centered at $y^0$, then the diameter of the convex hull of $Y$ is given by:

$$\Delta = \Delta(Y) = \max_i \|y^i - y^0\|$$

Therefore, for $k = 0$, the bound in (3.4) can be simplified to:

$$|m(x) - f(x)| \leq \frac{p+1}{(d+1)!} G \Lambda_Y \Delta^{d+1} \tag{3.5}$$

where

$$\Lambda_Y := \max_{0 \leq i \leq p} \max_x |\ell_i(x)|$$

This means $G$ depends only on $f$, and $\Lambda_Y$ depends only on $Y$. Moreover, since $\Delta(Y)$ is the trust region radius, we need to ensure the right hand side of (3.5) goes to 0 as $\Delta(Y)$ vanishes for the convergence of the Derivative Trust Region method. That is, we want the following relationship:

$$\Delta(Y) \to 0 \Longrightarrow (p+1) \cdot \sum_{i=0}^{p} \|y^i - x\|^{d+1} \to 0 \Longrightarrow |m(x) - f(x)| \to 0$$

In other words, $\Lambda_Y$ will have to be uniformly bounded for all $Y$ in the algorithm.

### 3.2.2 $\Lambda$-poisedness

In this section we will discuss characteristics of $\Lambda_Y$, and we will discuss methods to construct a "well-poised" set.

Let us first formally define the notion of "well-poisedness" of $Y$ (Definition 3.2 [24]):

**Definition 3.9.** Given $\Lambda \geq 1$, let $\phi = \{\phi_0(x), \ldots, \phi_p(x)\}$ be a basis of $\mathcal{P}_n^d$. A poised set $Y = \left\{y^0, \ldots, y^p\right\} \subseteq B \subseteq \Omega$ is $\Lambda$-***poised*** in $B$ if for any $x \in B$, there exists $\lambda(x) \in \mathbb{R}^{(p+1)}$ such that

$$\phi(x) = \sum_{i=0}^{p} \lambda_i(x)\phi(y^i) \;\text{ where }\; \|\lambda(x)\| \leq \Lambda$$

where $\phi(x) = [\phi_0(x), \ldots, \phi_p(x)]^\top$ is a vector in $\mathbb{R}^{(p+1)}$.

***Equivalently*** [25]: if $\{\ell_i(x)\}_{i=0}^p$ is the basis of Lagrange polynomials corresponding to $Y$ (as in lemma 3.7), then $Y$ is $\Lambda$***-poised*** if:

$$\Lambda \geq \max_{0\leq i\leq p}\max_{x\in B}|\ell_i(x)|$$

In other words, if we replace any point $\bar{y} \in Y$ by any $x \in B$, the volume of $\phi(Y)$ changes by at most a factor of $\Lambda$.

*Remark* 3.10. Conn, Scheinberg and Vicente (Lemma 3.8,3.9 of [25]) showed that the constant $\Lambda$ defined above is independent of scaling and translation.

For the rest of the discussion, we may assume without lost of generality that the the smallest closed ball containing $Y = \{y^0, \ldots, y^p\}$ is $B(0, 1)$, the unit sphere centered at 0. This can be done by the following transformation:

$$\hat{Y} = \{0, \hat{y}^1, \ldots, \hat{y}^p\} := \left\{0, \frac{y^1 - y^0}{\Delta(Y)}, \ldots, \frac{y^p - y^0}{\Delta(Y)}\right\} \subset B(0, 1) \tag{3.6}$$

We will now see how $\Lambda$-poisedness relates to $\kappa(\hat{M}) := \|\hat{M}\| \cdot \|\hat{M}^{-1}\|)$, the condition number of $\hat{M} := M(\bar{\phi}, \hat{Y})$, where $\bar{\phi}$ is the natural basis in $\mathcal{P}_n^d$ described in (3.3):

$$\bar{\phi} := \left\{1, x_1, x_2, \ldots, x_n, \frac{x_1^2}{2}, x_1x_2, \ldots, \frac{x_{n-1}^{d-1}x_n}{(d-1)!}, \frac{x_n^d}{d!}\right\}$$

Hence the Vandermonde matrix is given by:

$$\hat{M} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & \hat{y}_1^1 & \hat{y}_2^1 & \cdots & \hat{y}_n^1 & \frac{(\hat{y}_1^1)^2}{2} & \hat{y}_1^1\hat{y}_2^1 & \cdots & \frac{(\hat{y}_{n-1}^1)^{d-1}\hat{y}_n^1}{(d-1)!} & \frac{(\hat{y}_n^1)^d}{d!} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & \hat{y}_1^p & \hat{y}_2^p & \cdots & \hat{y}_n^p & \frac{(\hat{y}_1^p)^2}{2} & \hat{y}_1^p\hat{y}_2^p & \cdots & \frac{(\hat{y}_{n-1}^p)^{d-1}\hat{y}_n^p}{(d-1)!} & \frac{(\hat{y}_n^p)^d}{d!} \end{bmatrix} \tag{3.7}$$

Suppose $\hat{Y} \subset B(0, 1)$ is $\Lambda$-poised, by the first categorization of $\Lambda$-poisedness, for any $x \in B(0, 1)$, there exists $\lambda(x) \in \mathbb{R}^{(p+1)}$ such that

$$\bar{\phi}(x) = \sum_{i=0}^{p} \lambda_i(x)\bar{\phi}(\hat{y}^i) \text{ where } \|\lambda(x)\| \leq \Lambda$$

or equivalently

$$\bar{\phi}(x) = \hat{M}^\top\lambda(x) \text{ where } \|\lambda(x)\| \leq \Lambda$$

Since $\Delta(\hat{Y}) = 1$, $\exists \hat{y}^i \in \hat{Y}$ such that $\|\hat{y}^i\| = 1$, hence $\|\hat{M}\| \leq (p+1)^{\frac{3}{2}}$. Moreover, $x \in B(0,1)$ implies $\|\hat{M}\| \geq 1$, This implies $\|\hat{M}\|$ is bounded: $1 \leq \|\hat{M}\| \leq (p+1)^{\frac{3}{2}}$. Therefore to derive the relationship between $\kappa(\hat{M})$ and $\Lambda$, it suffices to determine the correlation between $\|\hat{M}^{-1}\|$ and $\Lambda$. The following theorem [25] provides us with an explicit bound:

**Theorem 3.11** (Theorem 3.14 of [25]). *Suppose $\hat{Y} \subset B(0,1)$ is a poised, and $\hat{M} := M(\bar{\phi}, \hat{Y})$ then the following holds*

1. *If $\hat{M}$ is non-singular, and $\|\hat{M}^{-1}\| \leq \Lambda$, then $\hat{Y}$ is $\sqrt{p+1} \cdot \Lambda$-poised in $B(0,1)$.*
2. *If $\hat{Y}$ is $\Lambda$-poised in $B(0,1)$, then $\hat{M}$ is non-singular, and $\|\hat{M}^{-1}\| \leq \theta \cdot (p+1)^{\frac{1}{2}}\Lambda$. The constant $\theta > 0$ is independent of $\hat{Y}$ and $\Lambda$, but is dependent on $n, d$.*

*Moreover $\theta$ is bounded above by 1, and $4 \cdot \sqrt{p+1}$ for $d = 1$ and $d = 2$ respectively*

*Remark* 3.12. Despite the fact that Vandermonde matrices are ill-conditioned, in practice the upper bound of the condition number of $\hat{M}$ is generally not very large, as the degree $d$ is typically small ($d = 1, 2$).

Theorem 3.11 allows us to construct a bound of the error between the object function and the interpolation polynomial directly. In particular we will consider the case when $d = 2$, where $m$ is a quadratic interpolation of $f$ as discussed in [25]. Note that the result discussed in the following section can be generalized to polynomial interpolations of any degree $d$.

### 3.2.3 Error bound of quadratic interpolation model

Let $Y = \{y_0, \ldots, y^p\}$ be a poised set of interpolation points in $B(y^0, \Delta(Y))$, where $p + 1 = \binom{n+d}{d} \overset{(d=2)}{=} \frac{(n+1)(n+2)}{2}$.

Suppose we construct the scaled set of interpolation points $\hat{Y}$ as in (3.6), and partition the respective Vandermonde matrix $\hat{M}$ as follows:

$$\left[\begin{array}{c|c} 1 & \vec{0} \\ \hline \vec{e} & \hat{Q} \end{array}\right] := \left[\begin{array}{c|cccccccc} 1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \hline 1 & \hat{y}_1^1 & \hat{y}_2^1 & \cdots & \hat{y}_n^1 & \frac{(\hat{y}_1^1)^2}{2} & \hat{y}_1^1\hat{y}_2^1 & \cdots & \frac{(\hat{y}_{n-1}^1)^{d-1}\hat{y}_n^1}{(d-1)!} & \frac{(\hat{y}_n^1)^d}{d!} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & \hat{y}_1^p & \hat{y}_2^p & \cdots & \hat{y}_n^p & \frac{(\hat{y}_1^p)^2}{2} & \hat{y}_1^p\hat{y}_2^p & \cdots & \frac{(\hat{y}_{n-1}^p)^{d-1}\hat{y}_n^p}{(d-1)!} & \frac{(\hat{y}_n^p)^d}{d!} \end{array}\right] = \hat{M} \tag{3.8}$$

where $\vec{e}$ denote the vector of all ones, and $\hat{Q}$ denote the lower right-hand submatrix $\hat{M}_{p\times p}$ of $\hat{M}$.

Before we state the error bound for the quadratic interpolation, let us first prove an exercise from [25]:

**Lemma 3.13** (Exercise 11 of [25]). *Suppose $\hat{M}$ and $\hat{Q}$ are defined as in (3.8) above, then*

$$\|\hat{Q}^{-1}\| \leq \|\hat{M}^{-1}\|$$

*Proof.*

$$\hat{M} = \begin{bmatrix} 1 & \vec{0} \\ \vec{e} & \hat{Q} \end{bmatrix}$$

By Schur complement we obtain $\hat{M}^{-1}$ in block matrix form::

$$\hat{M}^{-1} = \begin{bmatrix} (1-0)^{-1} & \vec{0} \\ -\hat{Q}^{-1}\vec{e}(1^{-1}) & \hat{Q}^{-1} \end{bmatrix} = \begin{bmatrix} 1 & \vec{0} \\ -\hat{Q}^{-1} & \hat{Q}^{-1} \end{bmatrix}$$

Hence

$$\|\hat{M}^{-1}\| \geq \|\hat{Q}^{-1}\|$$

□

The following theorem thus provides us with an error bound of quadratic interpolation model. As we have discussed, the following result can be generalized to polynomial interpolation models of any degree $d$:

**Theorem 3.14** (Theorem 3.16 of [25]). *Suppose $Y = \{y_0, \ldots, y^p\}$ is a poised in $B(y^0, \Delta(Y))$, and $f$ is continuously differentiable in an open set $\Omega$ containing $B(y^0, \Delta(Y))$, and $\nabla^2 f$ is Lipschitz continuous with Lipschitz constant $L > 0$. Then for any $y \in B(y^0, \Delta(Y))$ the following holds:*

1. *The error of the approximation of Hessian is bounded by:*

$$\|\nabla^2 f(y) - \nabla^2 m(y)\| \leq \kappa_H \Delta(Y)$$

*where*

$$\kappa_H = \frac{3 \cdot \sqrt{2}}{2} p^{\frac{1}{2}} L \|\hat{Q}^{-1}\|$$

2. *The error of the approximation of Jacobian is bounded by:*

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_J \Delta(Y)^2$$

*where*

$$\kappa_J = \frac{3 \cdot (1+\sqrt{2})}{2} p^{\frac{1}{2}} L \|\hat{Q}^{-1}\|$$

3. *The error of the approximation of function is bounded by:*

$$|f(y) - m(y)| \leq \kappa_f \Delta(Y)^3$$

*where*

$$\kappa_f = \frac{(6+9\cdot\sqrt{2})}{4} p^{\frac{1}{2}} L \|\hat{Q}^{-1}\| + \frac{L}{6}$$

### 3.2.4 Summary of Λ-poisedness results

If the set of interpolation points $Y = \{y^0, \ldots, y^p\}$ is Λ-poised, then by remark 3.10 $\hat{Y}$ is also Λ-poised. This implies, by theorem 3.11, that $\|\hat{M}^{-1}\|$ is bounded above by: $\|\hat{M}^{-1}\| \leq \theta \cdot (p+1)^{\frac{1}{2}} \Lambda$.

Combining this with lemma 3.13, we have:

$$\|\hat{Q}^{-1}\| \leq \|\hat{M}^{-1}\| \leq \theta \cdot (p+1)^{\frac{1}{2}} \Lambda < \infty$$

Therefore the constants $\kappa_H, \kappa_J$ and $\kappa_f$ of theorem 3.14 are all bounded above in terms of Λ, which implies the errors of approximation of the $i^{th}$ derivative are bounded by Λ and $\Delta(Y)^{3-i}$. This agrees with the error bound derived by Ciarlet and Raviart (Equation 3.4, Theorem 1, [30]).

### 3.2.5 Model Improvement Algorithms

During the execution of of an interpolation based Derivative Free Trust Region method, the set of interpolation points changes depending on the points reached, hence the corresponding polynomial interpolation also changes accordingly. To retain or improve the quality of the interpolation model, it is thus crucial to maintain the quality of the set of interpolation points.

In this section, we will discuss the algorithms from [25] to construct, maintain, and improve the poisedness of a given set of interpolation points.

Given a set of interpolation points $Y = \{y^0, \dots, y^q\}$ (not necessarily poised), and the corresponding basis of Lagrange polynomials $\{\ell_j(x)\} \subset \mathcal{P}_n^d$ from definition 3.6 (the Lagrange polynomials can be determined by Algorithm 1 of [25]):

$$\ell_j(y^i) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Suppose we want to replace a point $y^r \in Y$ by a new point $y^r_{new}$, we would have following new set of interpolation points:

$$Y^{new} = Y \setminus \{y^r\} \cup \{y^r_{new}\}$$

We will have one of the following cases depending on the value of $\ell_r(y^r_{new})$:

**If** $\ell_r(y^r_{new}) \neq 0$ **then**

The new set of Lagrange polynomials $\left\{\ell_j^{new}(x)\right\}$ can be constructed as follows [25]:

$$\textbf{Normalization: } \ell_r^{new}(x) = \frac{\ell_r(x)}{\ell_r(y^r_{new})}$$

$$\textbf{Orthogonalization: } \ell_j^{new}(x) = \ell_j(x) - \ell_j(y^r_{new})\ell_r^{new}(x), \text{for all} j \neq r$$

**Otherwise, if** $\ell_r(y^r_{new}) = 0$ **then**

Since the Lagrange polynomials $\{\ell_j(x)\}$ is a basis of $\mathcal{P}_n^d$, this implies corresponding matrix $M(\{\ell_j(x)\}, Y^{new})$ is singular, and the new set $Y^{new}$ is not poised.

Hence the set of Lagrange polynomials provides us with a natural way to construct a poised set from a given non-poised set $Y$. The following algorithm (Algorithm 2 of [25]) deals with the cases where $Y$ is either non-poised or $|Y| \leq \binom{n+d}{d}$.

For simplicity, let us denote $p$ by the integer such that

$$p + 1 := \binom{n+d}{d}$$

Note that if $|Y| < p + 1$, then $M(\phi, Y)$ cannot be non-singular for any basis $\phi$ of $\mathcal{P}_n^d$, and $Y$ must be non-poised. Therefore new points must be added to $Y$.

**Algorithm 2** From non-poised sets to poised sets

**INPUT:** A set of interpolation points $Y = \{y^0, \dots, y^q\}$ (not necessarily poised, $q+1 = |Y|$ not necessarily equal to $p+1$)

**OUTPUT:** A **poised** set of interpolation points $Y$, and the correspond basis of Lagrange polynomials $\{\ell_j(x)\}$

Initiate an approximation to the basis of Lagrange polynomials, the simplest example would be the monomial basis: $\{\ell_j(x)\} := \{\bar{\phi}_j(x)\}$.
**for** $i = 0, \dots, p$ **do**
  **Point Selection:** $j_i \leftarrow \operatorname{argmax}_{i \le j \le q+1} |\ell_i(y^j)|$
  **if** $|\ell_i(y^{j_i})| > 0$ and $i \le q+1$ **then**
    Swap $y^i$ and $y^{j_i}$ in $Y$
  **else**
    $y^i \leftarrow \operatorname{argmax}_{x \in B} |\ell_i(x)|$ ▷ $\ell_i(y^i) \neq 0$ as $\{\ell_i(x)\}$ is a basis
  **end if**
  **Normalization:** $\ell_i(x) \leftarrow \frac{\ell_i(x)}{\ell_i(y^i)}$
  **Orthogonalization:**
  **for** $j = 0, \dots, p$, $j \neq i$ **do**
    $\ell_j(x) \leftarrow \ell_j(x) - \ell_j(y^i)\ell_i(x)$
  **end for**
**end for**

*Remark* 3.15. The point selection step in the above algorithm above serves three purposes;

1. Includes all points in $Y$ that forms a poised set
2. Any point makes $Y$ non-poised will be discarded
3. If $|Y| < P$ then it will augment $Y$ by new points by determining $\operatorname{argmax}_{x \in B} |\ell_i(x)|$.

The normalization and orthogonalization step on the other hand constructs the Lagrange polynomial corresponding to the constructed poised set $Y$.

Now suppose we are given a poised set $Y$ in a closed ball $B$, the corresponding basis of Lagrange polynomials $\{\ell_i(x)\}$, and a constant $\Lambda > 1$. Recall from the second categorization of $\Lambda$- poisedness from definition 3.9: $Y$ is $\Lambda$-poised in $B \supset Y$ if:

$$\Lambda \ge \max_{0 \le i \le p} \max_{x \in B} |\ell_i(x)|$$

If $Y$ is not $\Lambda$-poised in $B$, then there exists index $i_k \in \{0, \dots, p\}$ such that:

$$\Lambda_{k-1} := \max_{x \in B} |\ell_{i_k}(x)| > \Lambda$$

In order to make $Y$ $\Lambda$-poised, we must replace the point $y^{i_k} \in Y$ by $y^{i_k}_{new} \in B$. In other words we update $Y$ by $Y^{new} = Y \setminus \{y^{i_k}\} \cup \{y^{i_k}_{new}\}$. The corresponding basis of

Lagrange polynomials must also be normalized and orthogonalized as in Algorithm 2, in particular:

$$\ell_{i_k}(x) \leftarrow \frac{\ell_{i_k}(x)}{\ell_{i_k}(y_{new}^{i_k})} \Longrightarrow \max_{x \in B} |\ell_{i_k}(x)| \leq 1 < \Lambda$$

The following algorithm (Algorithm 6.3 of [25]) constructs a $\Lambda$-poised of $Y$ by based the above arguments.

**Algorithm 3** From poised sets to $\Lambda$-poised sets

**INPUT:** A poised set $Y = \{y^0, \ldots, y^p\} \subset B$, the corresponding basis of Lagrange polynomials $\{\ell_i(x)\}$, and a constant $\Lambda > 1$

**OUTPUT:** A **$\Lambda$-poised** set of interpolation points $Y$ in $B$, and the correspond basis of Lagrange polynomials $\{\ell_j(x)\}$

$k \leftarrow 1$
**repeat**
  $\Lambda_{k-1} \leftarrow \max_{0 \leq i \leq p} \max_{x \in B} |\ell_i(x)|$
  **if** $\Lambda_{k-1} > \Lambda$ **then**
    $i_k \leftarrow \operatorname{argmax}_{0 \leq i \leq p} \max_{x \in B} |\ell_i(x)|$
    $y_{new}^{i_k} \leftarrow \operatorname{argmax}_{x \in B} |\ell_{i_k}(x)|$
    $Y \leftarrow Y \setminus \{y^{i_k}\} \cup \{y_{new}^{i_k}\}$
  **else**
    $\Lambda_{k-1} \leq \Lambda$ implies $Y$ is $\Lambda$-poised and stopping criterion is satisfied.
  **end if**
  $k \leftarrow k + 1$
  Compute and update the basis of Lagrange polynomials corresponding to $Y$
**until** Stopping criterion is satisfied

#### 3.2.5.1 Summary of model improvement algorithms

Algorithm 2 and algorithm 3 provide us with a natural and intuitive way to maintain the poisedness of the set of interpolation points during the execution of the Derivative Free Trust Region algorithm.

In light of the correspondence between $\Lambda$-poisedness of $Y$ and the condition number of $\hat{M} = M(\bar{\phi}, \hat{Y})$, authors of [24, 25] presented two alternative methods which factorizes $\hat{M}$ or $\hat{M}^\top$ with LU factorization or QR factorization respectively

It was shown in [25] that these alternative methods produces similar, if not better, results than algorithm 2 and algorithm 3. In particular, these alternative methods do not recompute the basis of Lagrange polynomials in the alternative methods and hence the overall complexity will be reduced.

However, these alternative methods require additional machinery, and hence are omitted for the purposes of our discussion. Detailed descriptions and analysis can be found in the cited papers.

For the remainder of the chapter, an execution of ***model improvement algorithms*** will be refereed to algorithms 2 if $Y$ is not poised, and algorithm 3 if $Y$ is poised.

### 3.2.6 Derivative Free Trust Region Algorithm (Quadratic Interpolation based)

In this section we will discuss the main algorithm of the quadratic interpolation based Derivative Free Trust Region method,and conditions that guarantees global convergence to second order local optima [25].

#### 3.2.6.1 Derivative Free Trust Region (Quadratic interpolation)

The main algorithm of Derivative Free Trust Region (Algorithm 10.2, 10.4, 11.2 of [25]) is as follows:

---

**Algorithm 4** DFO Trust Region method

---

**INPUT:** Initial solution $x_0 \in \Omega$, maximum radius $\Delta_{max} > 0$, initial trust region radius $\Delta_0 \in (0, \Delta_{max}]$, a set of interpolation points $Y_0$, and $m_0$ the set of Lagrange polynomials corresponding to $Y_0$

Constants: $\mu > \beta > 0$, $\omega \in (0,1)$, $\eta_1 \in (0,1)$, $\gamma_1 > 1 > \gamma_0 > 0$, $\epsilon_c > 0$, poisedness coefficient $\Lambda > 1$, poisedness improvement threshold $\Lambda_{imp}$, and an arbitrarily small stopping coefficient $\epsilon_s > 0$

**OUTPUT:** A local (possibly global) optimum that satisfies second order optimality conditions

$k \leftarrow 0$

$\breve{m}_0 \leftarrow m_0$; $\breve{g}_0 \leftarrow \nabla m_0$; $\breve{H}_0 \leftarrow \nabla^2 m_0$

$\breve{\sigma}_0^m \leftarrow \max\left\{\|\breve{g}_0\|, -\lambda_{min}(\breve{H}_0)\right\}$; $\breve{Y}_0 \leftarrow Y_0$

**repeat**

  **Criticality step:**

  **if** $\breve{\sigma}_k^m > \epsilon_c$ **then**

    $m_k \leftarrow \breve{m}_k$; $Y_k \leftarrow \breve{Y}_k$; $\Delta_k \leftarrow \breve{\Delta}_k$

  **else if** $\breve{\Delta}_k > \mu\breve{\sigma}_k^m$ **OR** $\breve{Y}_k$ is not $\Lambda$-poised in $B(x_k, \breve{\Delta}_k$ **then**

    Invoke ***criticality step*** with $(\Lambda, \omega, \mu, x_k, \breve{\Delta}_k, \breve{Y}_k)$ ▷ (†)

    Obtain $\tilde{m}_k$, and $\tilde{Y}_k$ $\Lambda$-poised in $B(x_k, \tilde{\Delta}_k)$ ▷ $\tilde{\Delta}_k \in (0, \mu\tilde{\sigma}_k^m]$

    $m_k \leftarrow \tilde{m}_k$; $Y_k \leftarrow \tilde{Y}_k$; $\Delta_k \leftarrow \min\left\{\max\left\{\tilde{\Delta}_k, \beta\tilde{\sigma}_k^m\right\}, \breve{m}_k\right\}$

  **else**

    $m_k \leftarrow \breve{m}_k$; $Y_k \leftarrow \breve{Y}_k$; $\Delta_k \leftarrow \breve{\Delta}_k$

  **end if**

  **Step computation:**

  Compute step $s_k$, and $x_k^+ \leftarrow x_k + s_k \in B(x_k, \Delta_k)$ ▷ (‡)

**Point acceptance:**

$$\rho_k = \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)})$$

**if** $\rho_k \geq \eta_1$ **then**

**Successful step:** $\breve{x}_{k+1} \leftarrow x_k^+$; increase or retain $\Delta_k$ by **radius update** step

Generate $\breve{Y}_{k+1}$ by applying model improvement algorithm to $Y_k \cup \{\breve{x}_{k+1}\}$ on $B(\breve{x}_{k+1}, \breve{\Delta}_{k+1})$

**else if** $\eta_1 > \rho_k \geq 0$ **AND** $Y_k$ is $\Lambda$-poised in $B(x_k, \Delta_k)$ **then**

**Acceptable step:** $\breve{x}_{k+1} \leftarrow x_k^+$, reduce $\Delta_k$ by **radius update** step

Generate $\breve{Y}_{k+1}$ by applying model improvement algorithm to $Y_k \cup \{\breve{x}_{k+1}\}$ on $B(\breve{x}_{k+1}, \breve{\Delta}_{k+1})$

**else if** $\eta_1 > \rho_k$ **AND** $Y_k$ is not $\Lambda$-poised in $B(x_k, \Delta_k)$ **then**

**Model improving step:** $\breve{x}_{k+1} \leftarrow x_k$

Generate $\breve{Y}_{k+1}$ by invoking **model improvement** below

**else if** $\rho_k < 0$ **AND** $Y_k$ is $\Lambda$-poised in $B(x_k, \Delta_k)$ **then**

**Unsuccessful step:** reduce $\Delta_k$ by **radius update** step, everything else remains unchanged, i.e. $\breve{x}_{k+1} \leftarrow x_k$ and $\breve{Y}_{k+1} \leftarrow Y_k$

**end if**

**Model improvement:**

**if** $\eta_1 > \rho_k$ **AND** $Y_k$ is not $\Lambda$-poised in $B(x_k, \Delta_k)$ **then**

**repeat**

Apply model improvement algorithms to $Y_k$ on $B(x_k, \Delta_k)$

**until** $Y_K$ is at least $(\Lambda + \Lambda_{imp})$-poised

$\breve{Y}_{k+1} \leftarrow Y_k$; $\breve{m}_{k+1}$ is the set of Lagrange polynomials corresponding to $\breve{Y}_{k+1}$

**end if**

**Trust region radius update:**

$$\breve{\Delta}_{k+1} \in \begin{cases} \{\min\{\gamma_1 \Delta_k, \Delta_{\max}\}\} & \text{for } \textbf{Successful step} \ \ \textbf{AND } \Delta_k < \beta \sigma_k^m \\ [\Delta_k, \min\{\gamma_1 \Delta_k, \Delta_{\max}\}] & \text{for } \textbf{Successful step} \ \ \textbf{AND } \Delta_k \geq \beta \sigma_k^m \\ \{\gamma_0 \Delta_k\} & \text{for } \textbf{Acceptable step} \\ \{\Delta_k\} & \text{for } \textbf{Model improving step} \\ \{\gamma_0 \Delta_k\} & \text{for } \textbf{Unsuccessful step} \end{cases}$$

$k \leftarrow k + 1$

**until** $\Delta_k < \epsilon_s$

(†): **Criticality step** invoked in the algorithm can be described as follows [25]:

Note that the following algorithm is invoked only if in the $k^{th}$ iteration:

1. $\breve{\sigma}_k^m \leq \epsilon_c$ **AND** one of the following holds:

2. • $\breve{\Delta}_k > \mu\breve{\sigma}_k^m$

   • $\breve{Y}_k$ is not $\Lambda$-poised in $B(x_k, \breve{\Delta}_k$

---

**Algorithm 5** Criticality step

**INPUT:** A set of interpolation points $Y$, a point $x \in \Omega$, trust region radius $\Delta$, and constants $\omega \in (0,1)$, $\mu > 0$, and poisedness coefficient $\Lambda > 0$.

**OUTPUT:** A $\Lambda$-poised set $\tilde{Y}$ in $B(x, \tilde{\Delta})$

1: $i \leftarrow 0$; let $m(x)$ denote the interpolation model associated with $Y$, and let $\sigma := \sigma_k^m$ be computed as in (3.9)
2: **repeat**
3: Apply model improvement algorithm to $Y$ on $B(x, \Delta^i)$
4: $\tilde{\Delta} \leftarrow \Delta^i$
5: $i \leftarrow i + 1$
6: **until** $\tilde{\Delta} \leq \mu(\sigma)^i$ **OR** $Y$ is $\Lambda$-poised in $B(x, \tilde{\Delta})$

---

(‡): **Step computation**: At the $k^{th}$ iteration, to compute the step $s_k$, we consider the local model problem known as the Trust Region Subproblem on $m_k(x) \in \mathcal{P}_n^d$:

$$s_k = \min_{s \in B(0,\Delta_k)} m_k(x_k + s)$$
$$\text{where } m_k(x_k + s) = m_k(x_k) + s^\top g_k + \frac{1}{2} s^\top H_k s^\top$$
$$\text{and } g_k := \nabla m_k(x_k);\ H_k := \nabla^2 m_k(x_k)$$

The Trust Region Subproblem is a well-studied problem in many literatures, and there are a handful of ways to generate possible solutions.

The most natural choice of the step direction $s_k$ would be the Cauchy step: the direction of steepest descent. The ***Cauchy step*** is $s_k^c$ is given by:

$$\begin{aligned} s_k^C &:= -t_k^C g_k \\ \text{where } t_k^C &:= \underset{t\geq 0, x_k - tg_k \in B(x_k,\Delta_k)}{\operatorname{argmin}} m_k(x_k - tg_k) \end{aligned}$$

The change of function value by taking the steepest descent direction is bounded below by:

**Theorem 3.16** (Theorem 10.1 of [25])**.**

$$m_k(x_k) - m_k(x_k + s_k^C) \geq \frac{1}{2}\|g_k\| \min\left\{\frac{\|g_k\|}{\|H_k\|}, \Delta_k\right\}$$

However, to ensure global convergence to local optima with second order optimality conditions, we must take care of the negative curvature of the interpolation function $m_k(x)$. Therefore we will have to consider, in addition to the steepest descent direction alone, the path of greatest negative curvature in the step generation:

Assume $\lambda_{min}(H_k) < 0$, where $\lambda_{min}(H_k)$ is the smallest eigenvalue of $H_k$, then the ***eigen step*** $s_k^E$ is the eigenvector of $H_k$ (or the principal direction) corresponding to $\lambda_{min}(H_k) < 0$ satisfying the following conditions [25]:

$$(s_k^E)^\top g_k \leq 0,\ \|s_k^E\| = \Delta_k,\ (s_k^E)^\top H_k (s_k^E)^\top = \lambda_{min}(H_k)\Delta_k^2$$

The change of function value by taking the eigen step is bounded below by:

**Theorem 3.17** (Lemma 10.2 of [25])**.**

$$m_k(x_k) - m_k(x_k + s_k^E) \geq -\frac{1}{2}\lambda_{min}(H_k)\Delta_k^2$$

For the convergence of the algorithm, the explicit Cauchy steps or eigen steps is not required; instead we are only required to obtain a step $s_k$ that improves the value of $m_k$ by a fraction of Cauchy or eigen step, in particular we would require the following to hold:

$$\begin{aligned} m_k(x_k) - m_k(x_k + s_k) &\geq \kappa\left[m_k(x_k) - \min\left\{m_k(x_k + s_k^C), m_k(x_k + s_k^E)\right\}\right] \\ &\geq \frac{\kappa}{2}\max\left\{\|g_k\| \min\left\{\frac{\|g_k\|}{\|H_k\|}, \Delta_k\right\}, -\lambda_{min}(H_k)\Delta_k^2\right\} \qquad (\star) \end{aligned}$$

where $\kappa \in (0, 1]$ is a constant, and the second inequality $(\star)$ is given by Theorem 3.16 and Theorem 3.17.

#### 3.2.6.2 Global convergence of Derivative Free Trust Region

To establish the convergence of the main algorithm, let us make the following additional assumptions about $f$ and the interpolation model $m_k(x)$ [25]:

1. Given an initial state $x_0$ and a maximum radius $\Delta_{max}$, $f$ is continuously differentiable, and Lipschitz continuous in $L_{enl}(x_0)$, where:

$$L(x_0) = \{x \in \mathbb{R}^n | f(x) \leq f(x_0)\}$$
$$L_{enl}(x_0) = \bigcup_{x \in L(x_0)} B(x, \Delta_{max})$$

2. $f$ is bounded below in $L(x_0)$.
3. $\|H_k\|$ is bounded above for all $k$, where $H_k$ denote the Hessian matrix of the polynomial approximation $m_k(x)$ in the $k^{th}$ iteration

Furthermore, we require a notion to measure the second order optimality condition of the interpolation model $m_k(x)$ at the $k^{th}$ iteration:

$$\sigma_k^m := \max\{\|g_k\|, -\lambda_{min}(H_k)\} \tag{3.9}$$

where $g_k = \nabla m_k(x)$, $H_l = \nabla^2 m_k(x)$, and $\lambda_{min}(H_k)$ is the smallest eigenvalue of $H_k$.

*Remark* 3.18. As $\sigma_k^m \to 0$ then both $\|g_k\| \to 0$, and $-\lambda_{min}(H_k) \to 0$, which implies $\nabla m_k(x_k) = 0$, and $\nabla^2 m_k(x_k)$ is positive definite respectively.

Conn, Scheinberg and Vicente [25] proved that, with the assumptions above, the quadratic interpolation based Derivative Free Trust Region method with step satisfying ($\star$) achieves second order limit-type global convergence to local optima:

**Theorem 3.19** (Theorem 10.24 of [25]). *Suppose the above assumptions hold, then*

$$\lim_{k \to \infty} \sigma_k = 0$$
$$\textit{where } \sigma_k := \sigma_k^f = \max\left\{\|\nabla f(x_k)\|, -\lambda_{min}(\nabla^2 f(x_k)\right\}$$

*By remark 3.18, the main algorithm converges to a local optimum satisfying second order optimality conditions*

This theorem suggests that Derivative Free Trust Region method only guarantee global convergence to local optima, similar to the classical Trust Region method.

In the following section we will discuss ways to escape local optima withe technique borrowed from Simulated Annealing.

## 3.3 Escaping local optima with Metropolis criterion

Recall from section 2.5.1, Simulated Annealing method escapes local optima by accepting uphill moves that could potentially worsen the objective function value. An acceptance rule is employed depending on the adaptation of the algorithm, to decide whether uphill moves should be accepted

Thus in this section, an acceptance rule is incorporated to enable acceptance of uphill moves. In particular, we will incorporate Metropolis criterion into the point acceptance step:

**Point Acceptance with Metropolis Criterion**

Compute $\rho_k$:

$$\rho_k = \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)})$$

**if** $\rho_k \geq \eta_1$ **then**
  **Successful step:** $\breve{x}_{k+1} \leftarrow x_k^+$; increase or retain $\Delta_k$ by **radius update** step
  Generate $\breve{Y}_{k+1}$ by applying model improvement algorithm to $Y_k \cup \{\breve{x}_{k+1}\}$ on $B(\breve{x}_{k+1}, \breve{\Delta}_{k+1})$
**else if** $\eta_1 > \rho_k \geq 0$ **AND** $Y_k$ is $\Lambda$-poised in $B(x_k, \Delta_k)$ **then**
  **Acceptable step:** $\breve{x}_{k+1} \leftarrow x_k^+$, reduce $\Delta_k$ by **radius update** step
  Generate $\breve{Y}_{k+1}$ by applying model improvement algorithm to $Y_k \cup \{\breve{x}_{k+1}\}$ on $B(\breve{x}_{k+1}, \breve{\Delta}_{k+1})$
**else if** $\eta_1 > \rho_k$ **AND** $Y_k$ is <u>not</u> $\Lambda$-poised in $B(x_k, \Delta_k)$ **then**
  **Model improving step:** $\breve{x}_{k+1} \leftarrow x_k$
  Generate $\breve{Y}_{k+1}$ by invoking **model improvement** below
**else if** $\rho_k < 0$ **AND** $Y_k$ is $\Lambda$-poised in $B(x_k, \Delta_k)$ **then**
  **Unsuccessful step:**
  Generate $p \in U[0,1]$; $-\Delta f \leftarrow f(x_k) - f(x_k^+)$
  **if** $\exp\left(\frac{-\Delta f}{\xi \cdot \Delta_k}\right) < p$ **then**
    $x_k^+$ treated as an **acceptable step**:
    $\breve{x}_{k+1} \leftarrow x_k^+$, reduce $\Delta_k$ by **radius update** step
    Generate $\breve{Y}_{k+1}$ by applying model improvement algorithm to $Y_k \cup \{\breve{x}_{k+1}\}$ on $B(\breve{x}_{k+1}, \breve{\Delta}_{k+1})$
  **else**
    Reduce $\Delta_k$ by **radius update** step, everything else remains unchanged, i.e. $\breve{x}_{k+1} \leftarrow x_k$ and $\breve{Y}_{k+1} \leftarrow Y_k$
  **end if**
**end if**

*Remark* 3.20. We have modified the **unsuccessful step** by the Metropolis criterion:

$$P_M = \min\left\{1, \exp\left(\frac{-\Delta f}{\xi \cdot \Delta_k}\right)\right\}$$
$$\text{where } -\Delta f = f(x_k) - f(x_k^+), \text{ and } \xi \in \mathbb{R}_{++}$$

This will only occur when $Y_k$ is $\Lambda$-poised in $B(x_k, \Delta_k)$ and $\rho_k < 0$ ($\Leftrightarrow \Delta f > 0$). Furthermore, as $k \to +\infty$, $\Delta_k \to 0$ (lemma 10.20 of [25]), we can replace temperature function $t_k$ by a scaled trust region radius $\xi \cdot \Delta_k$. The scaling constant $\xi \in \mathbb{R}_{++}$ will thus determine the initial temperature and the rate of cooling.

This allows us to accept uphill moves with probability $P_M = \exp\left(\frac{-\Delta f}{\xi \cdot \Delta_k}\right)$, which vanishes as $\xi \cdot \Delta_k \to 0$.

In the following section, we will address the shortcomings of Simulated Annealing by utilizing the search direction $s_k$ in determing a "good" neighborhood function $\mathcal{N}$ of the Simulated Annealing algorithm.

## 3.4 Topological Annealing

In this section we will address the insensitivity of Simulated Annealing towards topological information. We present a new approach, Topological Annealing, which utilizes the search direction $s_k$ generated by the Derivative Free Trust Region method described in the previous sections to generate neighborhoods.

Topological information can be incorporated into Simulated Annealing in different forms with different behaviors depending on the Derivative Free Optimization method used to generate the local search direction. We will demonstrate below the bidirectional version using search direction $s_k$ generated by Derivative Free Trust Region method.

Recall from section 3.2.6: when given an unconstrained optimization problem with objective function $f$, for each iteration $k$ the algorithm finds a quadratic polynomial interpolation $m_k$ on a $\Lambda$-poised interpolation set $Y_k$. And from the interpolation model $m_k$ we determine a suitable step $s_k$ by solving the Trust Region Subproblem on $m_k$:

$$s_k = \min_{s \in B(0,\Delta_k)} m_k(x_k) + s^\top g_k + \frac{1}{2} s^\top H_k s^\top$$
$$\text{where } g_k := \nabla m_k(x_k);\ H_k := \nabla^2 m_k(x_k)$$

Furthermore, it was shown that, if $s_k$ contains a fraction of improvement of the Cauchy step and eigen step, then the algorithm converges globally to a local (global) optima

with second order optimality condition, therefore the step $s_k$ should satisfy:

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa \left[ m_k(x_k) - \min \left\{ m_k(x_k + s_k^C), m_k(x_k + s_k^E) \right\} \right] \qquad (\Diamond)$$

where $\kappa \in (0, 1]$.

As the Trust Region radius vanishes $\Delta_k \to 0$, so does the error $\|\nabla^i f - \nabla^i m\| \to 0$. Therefore we may assume, without lost of generality, that topology of the hyper-surface defined by the interpolation polynomial $m$ is a good approximate of the hyper-surface defined by $f$; and hence $s_k$ is a good approximation of the Trust Region Subproblem of $f$.

With this idea in mind, we turn our attention to the framework of the Simulated Annealing algorithm:

Suppose at the $k^{th}$ iteration of the Simulated Annealing algorithm, we are given a point $x_k \in \Omega \subseteq \mathbb{R}^n$. Without prior knowledge of the objective function $f$, we choose the neighborhood $N(x_k)$ to be an $(n-1)$-dimensional sphere of radius $\Delta_k := \Delta(t_k)$, where $t_k$ is the temperature parameter, i.e.

$$N(x_k) := B(x_k, \Delta_k)$$

Instead of choosing a new candidate point uniformly from $N(x_k)$, we compute the quadratic $(d = 2)$ interpolation based Trust Region search direction $s_k$ as follows:

1: Sample $p + 1 := \binom{n+2}{2}$ points $Y_k$ from $N(x_k)$
2: Apply model improvement algorithms to $Y$ until $Y$ is $\Lambda$-poised in $N(x_k)$
3: Generate quadratic polynomial interpolation $m \in \mathcal{P}_n^2$ of $f$ on $Y$.
4: Generate step $s_k$ by solving Trust Region Subproblem of $m$ such that the search direction $s_k$ satisfies the inequality $(\Diamond)$

*Remark* 3.21. For simplicity of the above algorithmic expression, the set of interpolation points $Y_k$ are re-sampled and readjusted with model improvement algorithms in each iteration. However, in practice we might wish to recycle the interpolation points $Y_k$ for the $(k+1)^{st}$ iteration.

Suppose we let $\vec{\theta_k} := \theta(s_k)$ be the angle of $s_k$ relative to an arbitrary fixed reference axis $x^r$ in some basis $\mathcal{B}$ of $\mathbb{R}^n$ centered at $x_k$. We generate the ellipsoidal neighborhood with the Trust Region search direction $s_k$ as follows:

1: Scale $N(x_k)$ by $\Delta_k + \|s_k\|$ in the $\pm x^r$ direction
2: Rotate $N(x_k)$ by $\vec{\theta_k}$ (align $\mathcal{B}$ with $s_k$)
3: Scale $N(x_k)$ by $\frac{1}{\Delta_k + \|s_k\|}$ in all directions

*Remark* 3.22. Scaling of directions (step 1 and 3) can be done by multiplication by diagonal matrices, hence the operations described above is a composition of diagonal action and rotation, which is an affine transformations on $N(x_k)$, which preserves the geometry of the neighborhood.

This modification of the neighborhood $N(x_k)$ allows us to incorporate topological information of the objective function $f$ into Simulated Annealing. As we search for new candidate points uniformly on the modified neighborhood, we take larger steps bias towards local optima that satisfies second order optimality conditions, and at the same time larger steps away from such local optima to avoid being trapped.

On the other hand, this also allows us to take smaller steps that are not directed towards any local optima, thus increasing the efficiency of the overall algorithm.

*Remark* 3.23. Similar approaches can be applied with any Derivative Free Optimization method that generates reasonable search directions.

For instance, we can utilize directional search methods, such as Generalized Pattern Search [25], which generates a set of $(n+1)$ to $2n$ possible directions with a positive spanning basis $D_k$ and chooses one $d \in D_k$ as the local search direction.

Positive spanning bases $D_k$ guarantee an existence of a descent direction of the objective function $f$ (Theorem 2.3 (iv) [25]) as there exists $d \in D_k$ such that $-\nabla f^\top d > 0$. Therefore in each iteration of Simulated Annealing, instead of generating a random point from a neighborhood of $x_k$, we can choose a new candidate point uniformly from the finite set $\{x + d \,|\, d \in D_k\}$.

*Remark* 3.24. It is worth noting that, whilst Topological Annealing with Trust Region direction searches for new points from an infinite set per iteration, it is biased towards second order optimum. Topological Annealing with positive spanning bases, on the other hand, searches for new points within a finite set per iteration and is only biased towards first order optimum.

# Chapter 4

# Conclusion

In this research paper, we presented and discussed properties of two major algorithms, namely Simulated Annealing (SA) and Derivative Free Trust Region (DFTR) to solve global optimization problems using only the values of the objective function.

Using techniques from DFTR method, and the quadratic interpolation of the objection function, we can generate search directions that leads us to second order local optima. By incorporating hill climbing techniques from SA, we allow DFTR to bypass local optima by accepting points that might worsen the objective function value.

We discuss the insensitivity of Simulated Annealing towards topological information and present a new approach, the Topology Annealing (TA) which incorporates modifications that takes into consideration topological information using Trust Region search direction to generate neighborhood function.

TA comes in different flavors depending on the Derivative Free Optimization method used to generate search direction, and is theoretically effective in solving for global optimum in continuous optimization problem with strong non-linearity. Future research will include implementation of TA, and further enhancements of TA by incorporating the notion of positive spanning basis $D_{\oplus}$.

# Appendix A

# Single Vessel Loading Problem

In this Appendix, we will show how we can adapt the components of Simulated Annealing to solve real life global optimization problems involving "Black-box" model.

## A.1 Problem Description: Single Vessel loading problem

A container port terminal is a facility that stores, handles, and transfers containers to various container vehicles and vessels for onward transportation. Containers placed in the yard of the port terminal are organized into blocks, and each block is further partitioned into stacks consisting of column of containers. This is illustrated in Fig A.1 below.

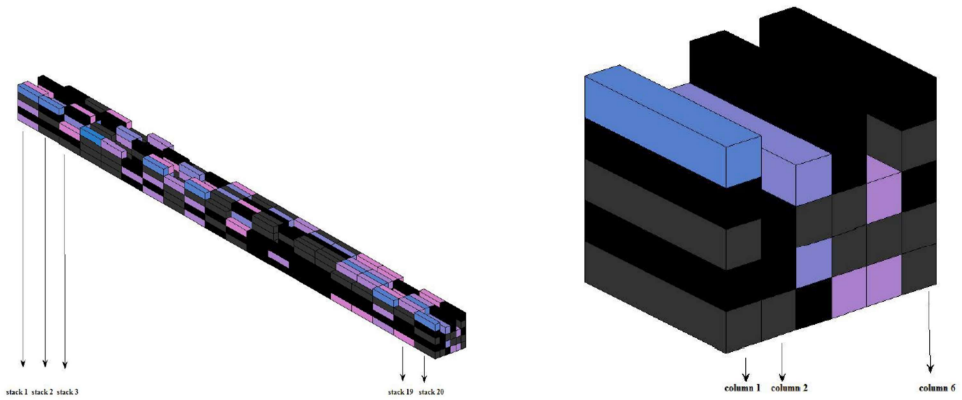


FIGURE A.1: The figure on the left shows an example of a block in the yard, whereas the figure on the right is a stack of containers within a block. Containers are sorted into different categories, which are represented by different colors.

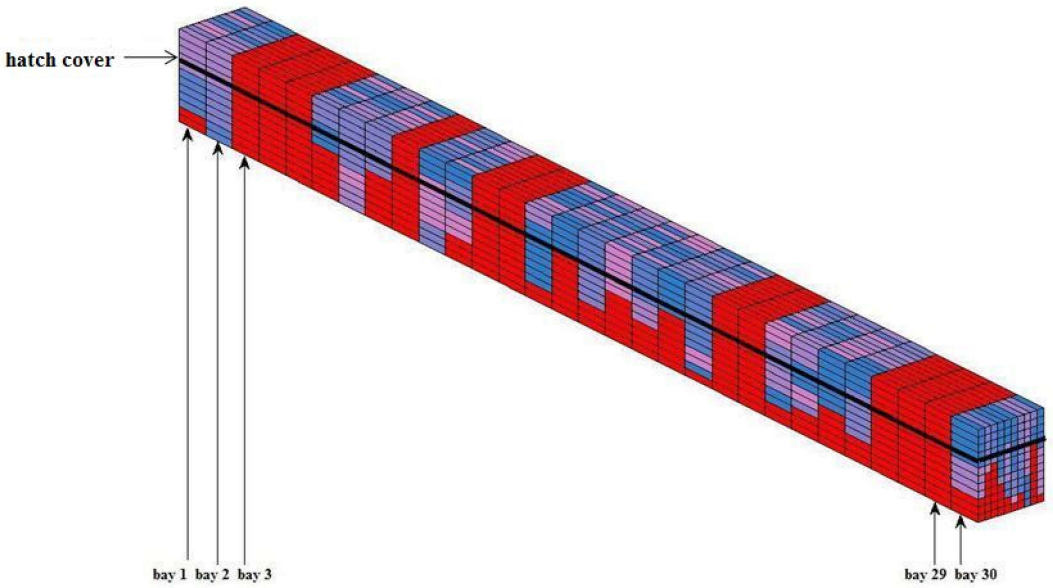


FIGURE A.2: This is a simplified representation of a container vessel's loading plan. Once again containers are sorted into different categories, and are represented by different colors.

Prior to entering the port terminal, container vessel provides the port terminal with a list of containers to be loaded onto the vessel also known as the *vessel loading plan.* Fig A.2 is a simplified representation of a loading plan. The port terminal then generates an *extraction list*, a list of containers in the yard to be loaded on to the vessel that matches the specification of the loading plan. The objective is to determine the most efficient way for the port terminal to load a single vessel.

### A.1.1 Definitions and Assumptions

Given a vessel and its vessel loading plan, let $\Omega$ be the set of extraction lists that satisfies the vessel loading plan. i.e. $w \in \Omega$ is a sequence of containers from the yard that maps bijectively onto the vessel loading plan.

Since containers are organized into blocks of stacks in the yard, we cannot physically move a container when there are other containers stacked on top of it, and hence we have the following definition:

**Definition A.1.** A container extraction move is called a ***productive move*** if the container is loaded directly from its stack to the vessel. Otherwise it is called a ***non-productive move***

**Example A.1** (Non-productive move)**.** *The following figure illustrates an example of non-productive moves. Containers #258 highlighted in the figure below can only be loaded by first removing the three #516 containers on top, resulting in three non-productive moves when relocating the three #516 containers.*


FIGURE A.3: The figure on the left shows the stack prior to the loading move, and the figure on the right shows the stack after the loading move. Notice the three #516 containers are moved out of the way but still remains in the stack

**Example A.2** (Productive move)**.** *The following figure illustrates an example of a productive move. Container#258 can be loaded onto the vessel directly where no non-productive move is required.*


FIGURE A.4: The figure on the left shows the stack prior to the loading move, and the figure on the right shows the stack after the loading move. Since no containers are in the way, we can load container #258 straight into the vessel.

Therefore given a vessel loading plan, we may conclude that the "most efficient way to load a vessel" is equivalent to determining an extraction list such that the number of

non-productive moves is minimized. Hence we define the objective function as follows:

$$\begin{aligned} f : \Omega &\to \mathbb{N} \\ w &\mapsto \text{Non-productive moves of } w \end{aligned}$$

In other words we would like to solve the following optimization problem:

$$\begin{aligned} &\min \{\text{Total number of non-productive moves of extraction list}\} \\ &\text{such that Containers in extraction list satisfies the vessel loading plan} \end{aligned}$$

or equivalently:

$$\begin{aligned} &\min f(w) \\ &\text{s.t.} w \in \Omega \end{aligned} \tag{†}$$

Furthermore, we will make the following assumptions:

1. Containers are classified into different categories; containers that belong to the same category are interchangeable
2. There is an existing black-box algorithm to compute the objective function $f : \Omega \to \mathbb{N}$ for any given $w \in \Omega$.
3. There is an existing black-box algorithm to determine a feasible extraction list $w_0 \in \Omega$, which will act as the initial guess.

Moreover, we would assume extraction lists $w \in \Omega$ are represented by sequences of containers of the following form:

$$w = \underbrace{\{x_1, x_2, \ldots\}}_{\text{each } x_i \text{ represent a container}},$$

$$\text{where } x_i := \{\text{yard location, category, bay location}\}$$

## A.2 Simulated Annealing Configuration

To solve (†) with Simulated Annealing, we must set up the components of Algorithm 1 describe in Chapter 2, section 2.3:

**Acceptance Function:**

For simplicity, we adopt the Metropolis criterion as the acceptance function:

$$P_m = A(x_k, x_{new}, t_k) = \min\left\{1, \exp\left(\frac{-\Delta f}{t_k}\right)\right\}$$

**Temperature Schedule:**

Set $t_0 = \infty$, define the cooling schedule as the exponential schedule:

$$t_k = U_k = \exp\left(-\frac{n}{\alpha}\right)$$

where $\alpha$ is a control parameter adjustable depending on the size of the problem. NOte that a repetition schedule is not employed in this case, therfore$U_k^m = U_k$.

**Neighborhood function**

To maintain feasibility of extraction list, the sequence of category extracted must remain the same. Hence the neighborhood function $\mathcal{N}(w)$ is chosen as follows:

$$\begin{aligned}\mathcal{N} : \Omega &\to \mathcal{D} \subseteq \mathcal{P}(\Omega)\\ w &\mapsto \{N(w)\} = \mathcal{N}(w)\end{aligned}$$

where for each extraction list $y \in N(w)$, the sequence of container categories of $y$ must be the same as that of $w$. In particular, we can move from $w$ to $y \in N(w)$ by:

1. Pick a container in $x$ in $w$
2. Obtain the container's category $c(x)$
3. Find other containers in $w$ of the same category, i.e. determine the set $C(x) := \{\bar{x} \in w \,|\, c(\bar{x}) = c(x)\} \subset w$
4. Shuffle and replace the original set $C(x)$ in $w$

The amount of change of neighborhood is completely determined by the size of $C(x)$, and therefore $|C(x)|$ is controlled by the temperature function. In this experiment we set: $|C(x)| = \max\{\lceil \beta \cdot t_k \rceil, 1\}$, where $\beta$ is another control parameter adjustable depending on the size of the problem.

In step 4 above, points in $C(x)$ are chosen uniformly, which also serves as the candidate distribution.

### A.2.1 Experimental Results

The following diagram is a simplified version with 26 containers. Two categories have been highlighted to indicate the precise swaps performed between the iterations

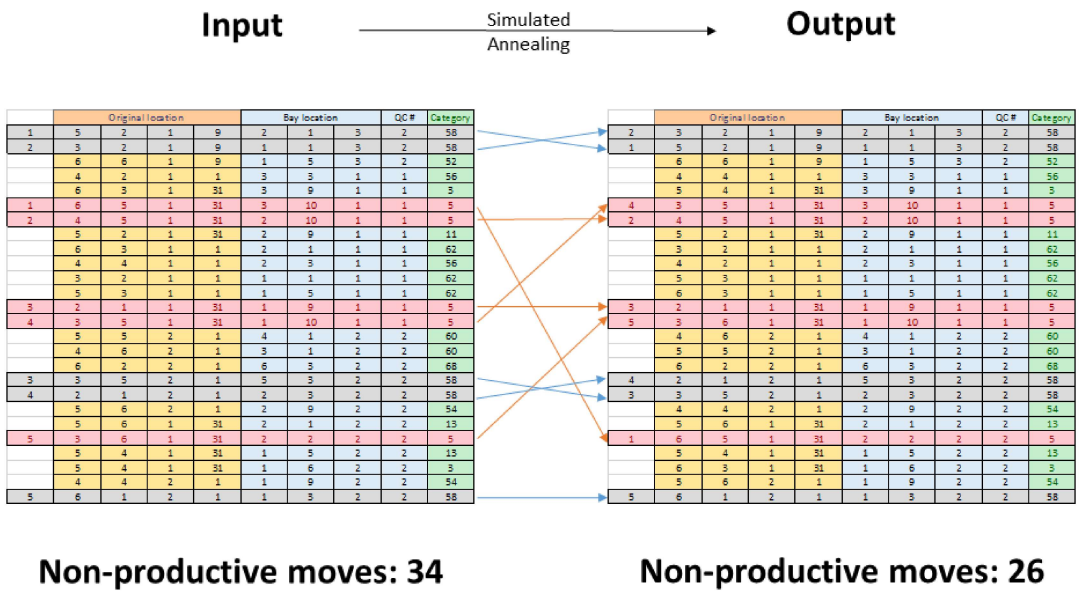


FIGURE A.5: Simplified input and output of Simulated Annealing applied to Single Vessel loading problem

The resulting extraction list is clearly feasible, and has a better objective function value.

In practice loading of a single vessel normally involve a large number of containers and the extraction list gets pretty large. In the following experiment we look into a bigger extraction list with 226 containers, where $\alpha = 650, \beta = 4$. The horizontal line represents the objective function value of the initial guess, and accepted solutions are labeled by a circle $\circ$, while rejected solutions are also displayed.

Note that each step of the annealing chain exhibits the behavior of a pure random search, and it also exhibits the behavior of a Boltzmann distribution.

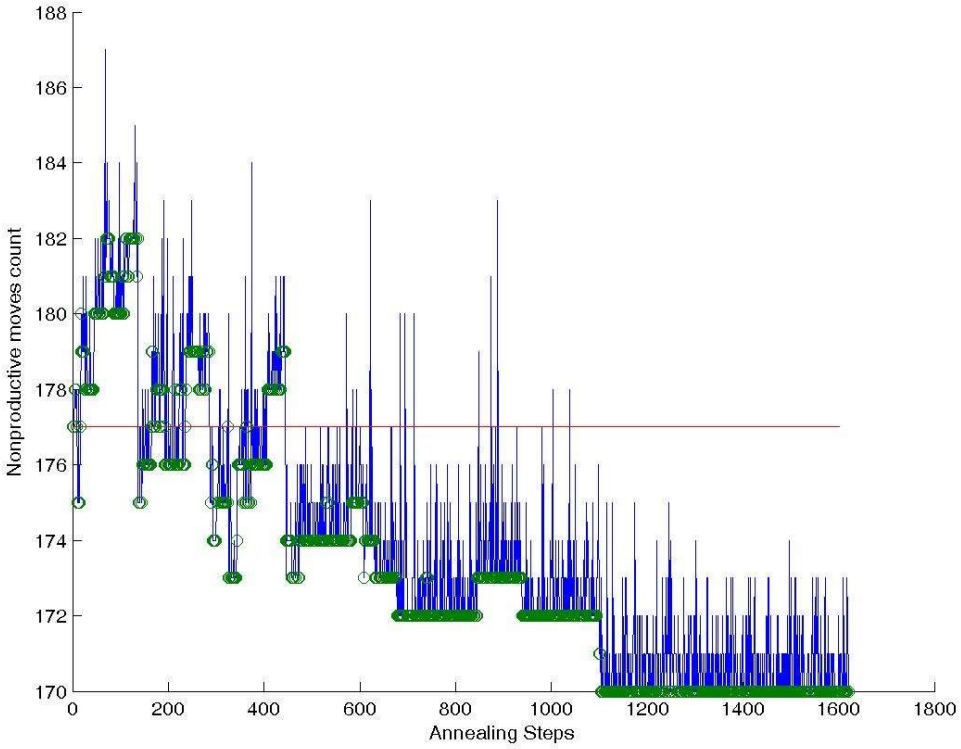


FIGURE A.6: $\alpha = 650, \beta = 4$

# Bibliography


[1] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953. doi: http://dx.doi.org/10.1063/1.1699114. URL http://scitation.aip.org/content/aip/journal/jcp/21/6/10.1063/1.1699114.

[2] R. W. Eglese. Simulated annealing: A tool for operational research. *European Journal of Operational Research*, 46(3):271–281, June 1990. URL http://ideas.repec.org/a/eee/ejores/v46y1990i3p271-281.html.

[3] Darrall Henderson, SheldonH. Jacobson, and AlanW. Johnson. The theory and practice of simulated annealing. In Fred Glover and GaryA. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, pages 287–319. Springer US, 2003. ISBN 978-1-4020-7263-5. doi: 10.1007/0-306-48056-5_10. URL http://dx.doi.org/10.1007/0-306-48056-5_10.

[4] M. Locatelli. Simulated annealing algorithms for continuous global optimization, 2000.

[5] A Zhigljavsky A, Žilinskas. Stochastic global optimization. *Springer Optimization and Its Applications*, 9:115–122, 2008. URL https://archive.org/details/towardextrapolat00davirich.

[6] Peter Salamon, Richard Frost, and Paolo Sibani. *Facts, Conjectures, and Improvements for Simulated Annealing*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. ISBN 0898715083.

[7] E Çinlar. Introduction to stochastic processes. page 153, 1974.

[8] Davis T.E. Toward an extrapolation of the simulated annealing convergence theory onto the simple genetic algorithm. 1991. URL https://archive.org/details/towardextrapolat00davirich.

[9] D. Mitra, F. Romeo, and A Sangiovanni-Vincentelli. Convergence and finite-time behavior of simulated annealing. pages 761–767, Dec 1985. doi: 10.1109/CDC.1985.268600.

[10] Bruce Hajek. Cooling schedules for optimal annealing. *Math. Oper. Res.*, 13(2): 311–329, May 1988. ISSN 0364-765X. doi: 10.1287/moor.13.2.311. URL http://dx.doi.org/10.1287/moor.13.2.311.

[11] Bertsimas D. and Tsitsiklis J. Simulated annealing. *Statistical Science*, 8:10–15, 1993. URL http://projecteuclid.org/euclid.ss/1177011077.

[12] Harry Cohn and Mark Fielding. Simulated annealing: Searching for an optimal temperature schedule. *SIAM J. on Optimization*, 9(3):779–802, March 1999. ISSN 1052-6234. doi: 10.1137/S1052623497329683. URL http://dx.doi.org/10.1137/S1052623497329683.

[13] Astrid Franz, Karl Heinz Hoffmann, and Peter Salamon. Best possible strategy for finding ground states. *Physics Review Letters*, 86:5219–5222, Jun 2001. doi: 10.1103/PhysRevLett.86.5219. URL http://link.aps.org/doi/10.1103/PhysRevLett.86.5219.

[14] H. Szu and R. Hartley. Fast simulated annealing. *Physics Letters A*, 122:157–162, June 1987. doi: 10.1016/0375-9601(87)90796-1.

[15] L. Ingber. Very fast simulated re-annealing. *Math. Comput. Model.*, 12(8):967–973, January 1989. ISSN 0895-7177. doi: 10.1016/0895-7177(89)90202-1. URL http://dx.doi.org/10.1016/0895-7177(89)90202-1.

[16] Gunter Dueck and Tobias Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *J. Comput. Phys.*, 90(1):161–175, August 1990. ISSN 0021-9991. doi: 10.1016/0021-9991(90)90201-B. URL http://dx.doi.org/10.1016/0021-9991(90)90201-B.

[17] P. Moscato and J. F. Fontanari. Stochastic versus deterministic update in simulated annealing. *Physics Letters A*, 146:204–208, May 1990. doi: 10.1016/0375-9601(90)90166-L.

[18] Ihor O Bohachevsky, Mark E Johnson, and Myron L Stein. Generalized simulated annealing for function optimization. *Technometrics*, 28(3):209–217, Aug 1986. ISSN 0040-1706. doi: 10.2307/1269076. URL http://dx.doi.org/10.2307/1269076.

[19] Philip N. Strenski and Scott Kirkpatrick. Analysis of finite length annealing schedules. *Algorithmica*, 6(1-6):346–366, 1991. ISSN 0178-4617. doi: 10.1007/BF01759050. URL http://dx.doi.org/10.1007/BF01759050.

[20] Anton Dekkers and Emile Aarts. Global optimization and simulated annealing. *Mathematical Programming: Series A and B*, 50(3):367–393, April 1991. ISSN 0025-5610. doi: 10.1007/BF01594945. URL http://dx.doi.org/10.1007/BF01594945.

[21] Pablo Moscato. An introduction to population approaches for optimization and hierarchical objective functions: A discussion on the role of tabu search. *Ann. Oper. Res.*, 41(1-4):85–121, May 1993. ISSN 0254-5330. URL http://dl.acm.org/citation.cfm?id=160231.160242.

[22] David Vanderbilt and Steven G Louie. A monte carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56 (2):259271, Nov 1984. doi: 10.1016/0021-9991(84)90095-0.

[23] Bülent Karasözen. Survey of trust-region derivative free optimization methods. *Journal of Industrial and Management Optimization*, 3(2):321–334, 2007. ISSN 15475816. doi: http://www.aimsciences.org/journals/displayArticles.jsp?paperID=2267.

[24] A.R. Conn, K. Scheinberg, and LusN. Vicente. Geometry of interpolation sets in derivative free optimization. *Mathematical Programming*, 111(1-2):141–172, 2008. ISSN 0025-5610. doi: 10.1007/s10107-006-0073-5. URL http://dx.doi.org/10.1007/s10107-006-0073-5.

[25] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009. ISBN 0898716683, 9780898716689.

[26] Virginia Torczon. On the convergence of pattern search algorithms. *SIAM J. on Optimization*, 7(1):1–25, January 1997. ISSN 1052-6234. doi: 10.1137/S1052623493250780. URL http://dx.doi.org/10.1137/S1052623493250780.

[27] M.J.D Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964. doi: 10.1093/comjnl/7.2.155. URL http://comjnl.oxfordjournals.org/content/7/2/155.

[28] M.J.D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In Susana Gomez and Jean-Pierre Hennart, editors, *Advances in Optimization and Numerical Analysis*, volume 275 of *Mathematics and Its Applications*, pages 51–67. Springer Netherlands, 1994. ISBN 978-90-481-4358-0. doi: 10.1007/978-94-015-8330-5_4. URL http://dx.doi.org/10.1007/978-94-015-8330-5_4.

[29] David Henry Winfield. Function and functional optimization by interpolation in data tables.

[30] P.G. Ciarlet and P.A. Raviart. General lagrange and hermite interpolation in rn with applications to finite element methods. *Archive for Rational Mechanics and Analysis*, 46(3):177–199, 1972. ISSN 0003-9527. doi: 10.1007/BF00252458. URL http://dx.doi.org/10.1007/BF00252458.