Lecture Notes for AE/ME 5222

Optimal Control of Dynamical Systems

Raghvendra V. Cowlagi

Assistant Professor, Aerospace Engineering Program,

Department of Mechanical Engineering, Worcester Polytechnic Institute.

January 2017

Chapter 1

Parameter Optimization

Optimal control theory involves the analysis and design of control algorithms to enable an optimal (i.e. “best” in whatever sense is appropriate for the problem) performance of a *process*, or, more precisely, a *dynamical system*. In general, these control algorithms will result in *functions* of an independent variable such as time. As a precursor to optimal control, we first study fundamental ideas in *parameter optimization*, which involves the selection of *static* quantities (i.e., numbers, not functions) that optimize a system’s performance. Table 1.1 provides examples that highlight the difference between optimal control problems and parameter optimization problems. Parameter optimization problems are usually referred to as simply *optimization* problems, or as *mathematical programming* problems, involve the following key elements:

1. The *decision variables*, which are the quantities whose values are to be determined for the system’s optimal performance.
2. The *domain*, which is the set of all acceptable values of the decision variables.
3. The *cost function* or *performance measure*, which is a scalar function that maps every element in the domain to a scalar value that represents the “goodness”’ of that element, i.e., the optimization problem is to find the minimum or maximum value(s) of the cost function.
4. *Constraints*, which are restrictions on the acceptable solutions to the problem. In general, constraints reduce the domain.

There are some fundamental concepts that must be remembered, as these will be applicable in optimal control theory as well.

1. An optimization problem need not necessarily have a solution. It is often important to establish the existence of solutions (or *feasibility*) before trying to find one.
2. A given cost function may have maximum or minimum values on one domain, but not another, i.e. the domain of optimization is an important component of the problem.
3. The solution to an optimization problem is not necessarily unique. The same problem can have several “equally good” solutions.
4. The inclusion or exclusion of constraints can affect the feasibility of optimization.

Table 1.1: Examples of problems in optimal control and parameter optimization.

Parameter Optimization	**Optimal Control**
Find the cruise speed to maximize the range of an aircraft.	Find a speed profile over a racing circuit for a car to minimize lap time.
Find the diameter of a cylindrical pipe to maximize flow rate.	Find the cross-sectional profile of a hypersonic aircraft nose section to minimize drag.
Find the minimum landing airspeed of an aircraft.	Find a landing trajectory for a fighter aircraft trying to land on a carrier deck.
Find the least expensive item in a restaurant menu that satisfies nutrition requirements.	Prepare a minimum-cost meal plan for a month that meets daily nutrition requirements and consists of up to three meals per day.

Example 1.1. *Find the largest number strictly less than 5.*

Note that this problem is ill-defined until we specify the domain, i.e., what is an acceptable "number". Suppose we say that the domain is $\mathbb{R}$, the set of all real numbers. Then this problem does *not* have a solution: pick any $x \in \mathbb{R}$ as close to 5 as desired, but strictly smaller. There is *always* another real number that is larger than x, but strictly smaller than 5 (why?).

Now suppose we specify the domain as $\mathbb{Z}$, the set of all integers. Then this problem has a unique solution: 4.

1.1 Unconstrained Optimization

Consider the case where the domain of optimization is $\mathbb{R}$, i.e., the decision variable, which we denote x is a scalar quantity that can take any real numerical value. We denote the cost function by $f : \mathbb{R} \to \mathbb{R}$, and we assume that the optimization problem is to find a scalar $x^* \in \mathbb{R}$ such that $f(x^*)$ is the minimum value of the cost function in its entire domain $\mathbb{R}$. There is no loss of generality here: if an application requires the maximization of a cost function, then the minimization of the negative of that cost function is an equivalent problem.

The problem stated above is, in general, difficult. A solution may not exist: consider for example $f(x) = x$. Even if a solution exists, finding the minimum value of f over the *entire* domain (also called the *global minimum*) can be difficult: often we will have to settle for a *local minimum*, which is a minimum value among "immediate neighbors", but not necessarily over the entire domain.

Theorem 1.1 (First-order Necessary Condition)**.** *If the function f attains a local minimum at $x^* \in \mathbb{R}$, and if the derivative $f'(x)$ exists at $x = x^*$, then $f'(x^*) = 0$.*

Proof. Because x^* is a local minimum, $f(x^*) \leqslant f(x^* + h)$ for every sufficiently small h, i.e. in the limit as $h \to 0$. Therefore,

$$\lim_{h \to 0^+} \frac{f(x^* + h) - f(x^*)}{h} \geqslant 0, \qquad \lim_{h \to 0^-} \frac{f(x^* + h) - f(x^*)}{h} \leqslant 0. \tag{1.1}$$

Because the derivative $f'(x)$ exists at $x = x^*$,

$$\lim_{h \to 0^+} \frac{f(x^* + h) - f(x^*)}{h} = \lim_{h \to 0^-} \frac{f(x^* + h) - f(x^*)}{h} = f'(x^*),$$

which implies, by Eqn. (1.1), that

$$f'(x^*) \geqslant 0, \qquad f'(x^*) \leqslant 0, \tag{1.2}$$

which can be true only if $f'(x^*) = 0$. □

Theorem 1.1 is a necessary condition for a local minimum, which means that any local minimum must satisfy the condition $f'(x^*) = 0$, but the satisfaction of this condition alone does not always mean that x^* is a local minimum. Points that satisfy necessary conditions for optimality, such as Theorem 1.1 are called *stationary* or *extremal* points.

Theorem 1.2 (Sufficient Condition)**.** *If $f'(x), f^{''}(x) \ldots, f^{(n)}(x)$ all exist and are continuous at the point $x = x^* \in \mathbb{R}$ for some integer $n \geqslant 1$, such that*

$$f'(x^*) = f^{''}(x^*) = \ldots = f^{(n-1)}(x^*) = 0, \tag{1.3}$$

and $f^{(n-1)}(x^*) \neq 0$, then the following statements are true:

(i) If n is even and $f^{(n)}(x^*) > 0$, then x^* is a local minimum,

(ii) If n is even and $f^{(n)}(x^*) < 0$, then x^* is a local maximum,

(iii) If n is odd, then x^* is neither a minimum nor a maximum.

Proof. The Taylor's series expansion of f about x^*, with an explicit formula for the n^{th}-order remainder, is:

$$f(x^* + h) = f(x^*) + hf'(x^*) + \frac{h^2}{2!}f^{''}(x^*) + \ldots + \frac{h^{n-1}}{(n-1)!}f^{(n-1)}(x^*) + \frac{h^n}{n!}f^{(n)}(x^* + ch),$$

where $0 < c < 1$. By Eqn. (1.3),

$$f(x^* + h) - f(x^*) = \frac{h^n}{n!}f^{(n)}(x^* + ch).$$

Because $f^{(n)}(x^*) \neq 0$, and because $f^{(n)}$ is continuous at x^*, there exists $\epsilon > 0$ such that for every $x \in [x^* - \epsilon, x^* + \epsilon]$, the sign of $f^{(n)}(x)$ is the same as the sign of $f^{(n)}(x^*)$. Therefore, in the limit $h \to 0$, the sign of $f^{(n)}(x^* + ch)$ is the same as that of $f^{(n)}(x^*)$. First suppose n is even. Then $h^n/n!$ is positive irrespective of the sign of h. Therefore, when $f^{(n)}(x^*)$ is positive, $f(x^* + h) - f(x^*)$ is positive irrespective of h, which means that x^* is a local minimum, which proves statement (i). Statement (ii) can be similarly proven. Now suppose n is odd. Then the sign of $h^n/n!$ depends on the sign of h, and hence $f(x^* + h) - f(x^*)$ changes sign whenever h changes sign. Therefore, x^* is neither a minimum nor a maximum, which proves statement (iii). □

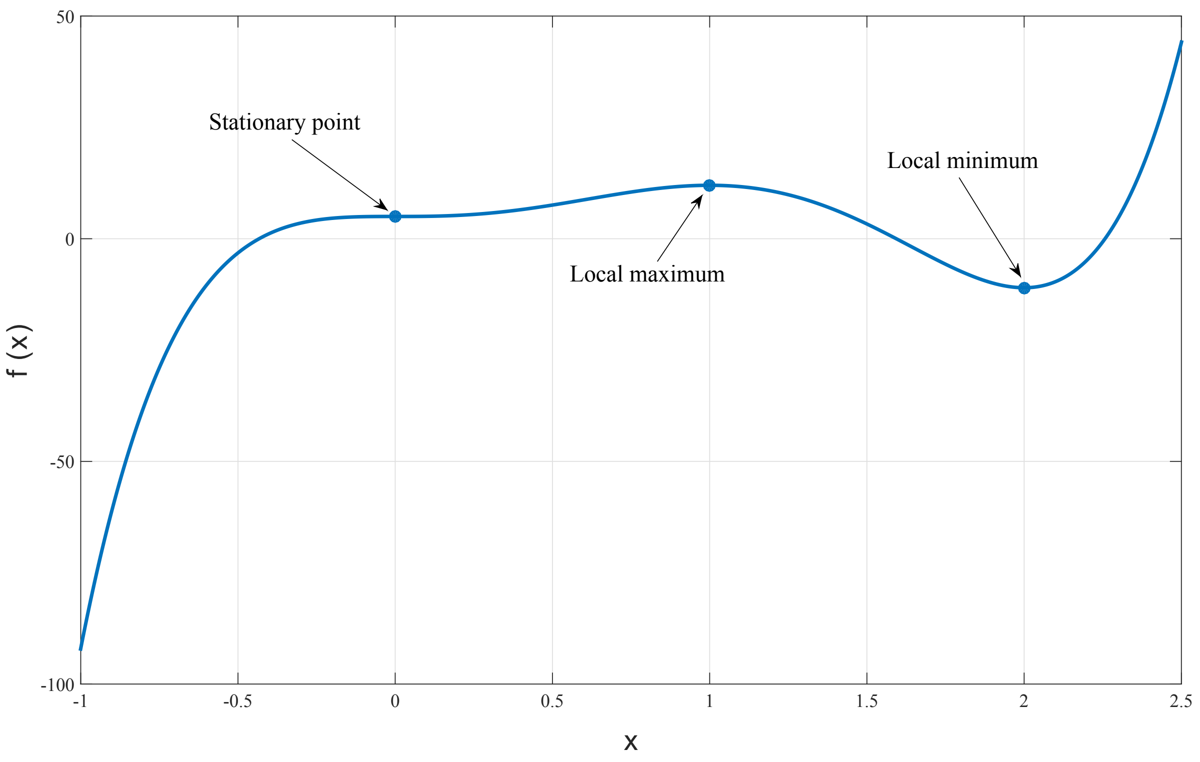


Figure 1.1: Visualization of the function f in Example 1.2.

Theorem 1.2 provides sufficient conditions, which means that if a point x^* satisfies all the stated conditions, then it is certainly a local minimum.

Example 1.2. Find the minimum value(s) of the function $f(x) := 12x^5 - 45x^4 + 40x^3 + 5$.

First, note that $f'(x) = 60(x^4 - 3x^3 + 2x^2) = 60x^2(x-1)(x-2)$, which means that $f'(x) = 0$ for $x = x_1^* = 0$, $x = x_2^* = 1$, $x = x_3^* = 2$. Next, note that $f''(x) = 60(4x^3 - 9x^2 + 4x)$.

At $x = x_2^*$, $f''(x) = -60$, which means, by Theorem 1.2, that x_2^* is a maximum, not a minimum. At $x = x_3^*$, $f''(x) = 240$, which means that x_3^* is a minimum. In particular, $f(x_3^*) = -11$. Finally, at $x = x_1^*$, $f''(x) = 0$, which means we must consider the signs of higher-order derivatives. Note that $f^{(3)}(x_1^*) = 240$. Because $f^{(3)}(x_1^*) \neq 0$, and 3 is odd, by Theorem 1.2, x_1^* is neither a minimum nor a maximum. Figure 1.1 shows a graphical visualization of the function f, with the three stationary points indicated. It is clear from Fig. 1.1 that $x_3^* = 2$ is only a local minimum, i.e., there are other points in the domain $\mathbb{R}$ where f attains lower values. Indeed, the function f is unbounded in either direction, therefore it has neither a global minimum nor a global maximum. $\square$

If the domain of optimization is a closed interval $[a_1, a_2] \subset \mathbb{R}$, where $a_1, a_2 \in \mathbb{R}$, and $a_2 > a_1$, then Theorems 1.1 and 1.2 are valid for all *interior* points of that interval, i.e., at all points in the interval $[a_1, a_2]$ *except* for the two boundary points a_1 and a_2. As evident in Fig. 1.1, a global minimum can lie on either of these boundary points, but will not be "detected" as a stationary point under the conditions of Theorems 1.1 and 1.2. This observation is at the heart of linear programming (i.e., the cost function and all constraints are linear), where the global optimum always lies on a boundary point, and the problem becomes one of finding the "right" boundary point out of several candidates.

Theorems 1.1 and 1.2 can be generalized for multivariable optimization: here, the domain is $\mathbb{R}^n$, where $n > 1$ is an integer, and the cost function $f : \mathbb{R}^n \to \mathbb{R}$ is a multivariable function. The *Hessian* matrix is defined as the symmetric square matrix of second-order partial derivatives of the function f:

$$\mathbf{H}(f)(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial {x_1}^2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{x}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial {x_2}^2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_n \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial {x_n}^2}(\mathbf{x}) \end{bmatrix}. \tag{1.4}$$

Theorem 1.3 (Necessary Condition). *If $\mathbf{x}^* \in \mathbb{R}^n$ is a stationary point of f, and if the first partial derivatives of f exist at $\mathbf{x}^*$, then*

$$\frac{\partial f}{\partial x_1}(\mathbf{x}^*) = \frac{\partial f}{\partial x_2}(\mathbf{x}^*) = \ldots = \frac{\partial f}{\partial x_n}(\mathbf{x}^*) = 0. \tag{1.5}$$

Theorem 1.4 (Sufficient Condition). *If all the first and second partial derivatives of f exist at $\mathbf{x}^* \in \mathbb{R}^n$, if Eqn. (1.5) is true, and if the Hessian matrix $\mathbf{H}(f)(\mathbf{x}^*)$ is positive definite, then $\mathbf{x}^*$ is a local minimum.*

A square matrix $Q \in \mathbb{R}^{n \times n}$ is said to be *positive definite* if $\mathbf{x}^{\mathrm{T}} Q \mathbf{x} > 0$ for every non-zero $\mathbf{x} \in \mathbb{R}^n$. This definition of positive definiteness is impractical to numerically verify, therefore the following result is typically used instead.

Proposition 1.1 (Bernstein, 2009). *A symmetric matrix $Q \in \mathbb{R}^{n \times n}$ is positive definite if and only if all of its eigenvalues are strictly positive.*

Example 1.3. Find the stationary points and minimum on $\mathbb{R}^2$ of the polynomial

$$f(x_1, x_2) = x_1^3 + x_2^3 + 2x_1^2 + 4x_2^2 + 6.$$

First, we check the necessary conditions stated in Theorem 1.3.

$$\frac{\partial f}{\partial x_1}(\mathbf{x}^*) = 3(x_1^*)^2 + 4x_1^* = 0, \qquad \frac{\partial f}{\partial x_2}(\mathbf{x}^*) = 3(x_2^*)^2 + 8x_2^* = 0.$$

Notice that these equations are "decoupled," i.e., the first equation consists of terms only in x_1 and the second equation consists of terms only in x_2. Therefore, it is easy to identify vectors $\mathbf{x}^*$ in $\mathbb{R}^2$ where $\frac{\partial f}{\partial x_1}(\mathbf{x}^*) = \frac{\partial f}{\partial x_2}(\mathbf{x}^*) = 0$. The solutions to the first equation are $0, -\frac{4}{3}$, whereas the solutions to the second equation are $0, -\frac{8}{3}$, and the stationary points of f in $\mathbb{R}^2$ are:

$$\mathbf{x}_1^* = (0, 0), \qquad \mathbf{x}_2^* = (0, -\frac{8}{3}), \qquad \mathbf{x}_3^* = (-\frac{4}{3}, 0), \qquad \mathbf{x}_4^* = (-\frac{4}{3}, -\frac{8}{3}).$$

To evaluate the sufficient condition in Theorem 1.4, we evaluate the Hessian matrix:

$$\mathbf{H}(f)(\mathbf{x}^*) = \begin{bmatrix} 6x_1 + 4 & 0 \\ 0 & 6x_2 + 8 \end{bmatrix}.$$

$$\Rightarrow \mathbf{H}(f)(\mathbf{x}_1^*) = \begin{bmatrix} 4 & 0 \\ 0 & 8 \end{bmatrix},$$

which is positive definite, and therefore we conclude that $\mathbf{x}_1^* = (0, 0)$ is a local minimum. Similarly, we can evaluate the Hessian at the other stationary points to find if it is positive definite. $\square$

1.2 Optimization with Equality Constraints

In this section, we introduce an additional element to the optimization problems considered so far: **equality constraints,** which specify certain relationships that the decision variables must satisfy. The general problem formulation is

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}),$$
$$\text{subject to } h(\mathbf{x}) = 0.$$

Here f is, as before, a scalar cost function, i.e., $f : \mathbb{R}^n \to \mathbb{R}$. The function h is, in general, a vector-valued function $h : \mathbb{R}^n \to \mathbb{R}^m$. Here m is the number of equality constraints. In other words, we have m scalar functions $h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_m(\mathbf{x})$, and

$$h(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_m(\mathbf{x}) \end{bmatrix}.$$

For a non-trivial optimization problem, $m < n$, for otherwise, we would be merely solving a set of simultaneous equations (i.e. nothing to optimize).

If the equation $h(\mathbf{x}) = 0$ is easy to solve analytically, then we can rewrite the above problem as an *unconstrained* problem in the reduced domain $\mathbb{R}^{n-m}$. However, this is usually impractical, and we need a more general technique to solve equality-constrained problems.

1.2.1 The Method of Lagrange Multipliers

Consider a function defined as follows:

$$L(\mathbf{x}, \mathbf{p}) = f(\mathbf{x}) + \mathbf{p}^{\mathrm{T}} h(\mathbf{x}), \tag{1.6}$$

where $\mathbf{p} = (p_1, p_2, \ldots, p_m) \in \mathbb{R}^m$ is an m-dimensional vector called the *Lagrange multiplier*. The function L is called a *Lagrangian function.* Written in scalar form:

$$L(\mathbf{x}, \mathbf{p}) = f(\mathbf{x}) + p_1 h_1(\mathbf{x}) + p_2 h_2(\mathbf{x}) + \ldots + p_m h_m(\mathbf{x}).$$

Now consider the problem of *unconstrained* minimization of the Lagrangian function L, i.e. we attempt to find vectors $(\mathbf{x}^*, \mathbf{p}^*) \in \mathbb{R}^{n+m}$ that minimize L. By Theorem 1.3, we have the following necessary conditions for this new minimization problem:

$$\frac{\partial L}{\partial x_i}(\mathbf{x}^*, \mathbf{p}^*) = 0, \text{ for each } i = 1, 2, \dots, n,$$
$$\frac{\partial L}{\partial p_j}(\mathbf{x}^*, \mathbf{p}^*) = 0, \text{ for each } j = 1, 2, \dots, m.$$

By the virtue of our definition of the function L, the second set of necessary conditions are:

$$\frac{\partial L}{\partial p_j}(\mathbf{x}^*, \mathbf{p}^*) = h_j(\mathbf{x}^*) = 0, \text{ for each } j = 1, 2, \dots, m.$$

Therefore, the necessary condition for the *unconstrained* minimization of L implies the satisfaction of the equality constraints in the original problem of constrained minimization of f. Furthermore, for all "acceptable" values of $\mathbf{x}$, i.e., whenever $h(\mathbf{x}) = 0$, the value of the Lagrangian is equal to the value of the cost function: $L(\mathbf{x}, \mathbf{p}) = f(\mathbf{x}) + \mathbf{p}^{\mathrm{T}} h(\mathbf{x}) = f(\mathbf{x})$, *irrespective of the values of the Lagrange multipliers* $\mathbf{p}$.

We conclude that the *unconstrained* minimization of L on the domain $\mathbb{R}^{n+m}$ is equivalent to the *constrained* minimization of f subject to the equality constraint h. Note that this makes the constrained minimization problem "easier," but at the expense of increasing the size of the domain.

It can be shown that a sufficient condition for $\mathbf{x}^* \in \mathbb{R}^n$ to be a constrained minimum of f is the Hessian $\mathbf{H}(L)(\mathbf{x}^*)$ must be positive definite, where the Hessian involves partial derivatives with respect to $\mathbf{x}$ only (and not with respect to $\mathbf{p}$).

Example 1.4. Find the dimensions of a cylinder of maximum volume with surface area 24π.

The decision variables are the radius x_1 of the cylinder base, and the height x_2 of the cylinder. Therefore, the domain is $\mathbb{R}^2$. The reward function is the cylinder volume, therefore the cost function f is the negative of this reward function:

$$f(x_1, x_2) = -\pi x_1^2 x_2.$$

There is one constraint ($m = 1$), namely, that the cylinder's surface area be equal to 24π, i.e., $2\pi x_1^2 + 2\pi x_1 x_2 = 2\pi(x_1^2 + x_1 x_2) = 24\pi$, which we rewrite in the standard form as:

$$\begin{aligned} h(x_1, x_2) &= x_1^2 + x_1 x_2 - 24\pi/2\pi \\ &= x_1^2 + x_1 x_2 - 12 = 0. \end{aligned}$$

Next, we write the Lagrangian function as

$$L(x_1, x_2, p) = f(x_1, x_2) + p h(x_1, x_2) = -\pi x_1^2 x_2 + p(x_1^2 + x_1 x_2 - 12),$$

where p is the Lagrange multiplier (scalar because $m = 1$). The necessary conditions for an unconstrained minimum $(\mathbf{x}^*, p^*)$ of the Lagrangian L are:

$$\frac{\partial L}{\partial x_1}(\mathbf{x}^*, p^*) = 0, \qquad \frac{\partial L}{\partial x_2}(\mathbf{x}^*, p^*) = 0, \qquad \frac{\partial L}{\partial p}(\mathbf{x}^*, p^*) = 0.$$

Evaluate the partial derivatives to arrive at the conditions:

$$\frac{\partial L}{\partial x_1}(\mathbf{x}^*, p^*) = -2\pi x_1^* x_2^* + p^*(2x_1^* + x_2^*) = 0, \tag{1.7}$$

$$\frac{\partial L}{\partial x_2}(\mathbf{x}^*, p^*) = -\pi (x_1^*)^2 + p^* x_1^* = 0, \tag{1.8}$$

$$\frac{\partial L}{\partial p}(\mathbf{x}^*, p^*) = (x_1^*)^2 + x_1^* x_2^* - 12 = 0. \tag{1.9}$$

Solving the first two equations, we arrive at two expressions for p^* :

$$p^* = \frac{2\pi x_1^* x_2^*}{2x_1^* + x_2^*}, \qquad\qquad p^* = \pi x_1^*.$$

We eliminate p^* to arrive at $x_1^* = \frac{1}{2} x_2^*$, which we can then use in (1.9) to arrive at

$$x_1^* = 2, \text{ and } x_2^* = 2x_1^* = 4.$$

The value of the Lagrange multiplier at this candidate minimum is $p^* = \pi x_1^* = 2\pi$. We need to verify that this solution is indeed a minimum by checking the positive definiteness of the Hessian.

Note: Try to solve this problem by first solving the constraint equation to express x_2 in terms of x_1, and then finding the unconstrained minimum of $f(x_1)$. □

Interpretation of the Lagrange multiplier

It can be shown that the Lagrange multiplier p indicates the sensitivity of the optimal value of the cost function to the constraint. In the preceding example, the positive value of the Lagrange multiplier p^* at the candidate minimum indicates that when the constraint changes from "area = 24π" to "area = $24\pi - \epsilon$," then the optimum value of the cost function increases, i.e., the volume of the cylinder *decreases* (remember that the cost here was the negative of the cylinder's volume, which we were trying to *maximize*).

If the Lagrange multiplier happens to be zero at a candidate minimum, then the implication is that the equality constraint has no effect, i.e., the candidate minimum will also be a candidate for if the problem were unconstrained.

1.3 Optimization with Inequality Constraints

In this section, we introduce another element to the general optimization problem considered so far, namely, **inequality constraints.** The general problem formulation is

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}),$$
$$\text{subject to } h(\mathbf{x}) = 0, \text{ and } g(\mathbf{x}) \leqslant 0.$$

Here f is, as before, a scalar cost function, i.e., $f : \mathbb{R}^n \to \mathbb{R}$. Similar to h, the function g is, in general, a vector-valued function $h : \mathbb{R}^n \to \mathbb{R}^\ell$. Here ℓ is the number of inequality constraints. In other words, we have ℓ scalar functions $g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_\ell(\mathbf{x})$, and

$$g(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ \vdots \\ g_\ell(\mathbf{x}) \end{bmatrix}.$$

Unlike the case with equality constraints, there can be any number of inequality constraints, i.e., there is no restriction on ℓ compared to n. An optimization problem with equality and inequality constraints is often referred to as a *mathematical program.*

To solve such problems, we often need to first find a vector $\mathbf{x}$ that simply satisfies all constraints (but does not necessarily minimize the cost function). Such a vector is called a *feasible* solution to the problem, and is often used as an initial "guess" to kickstart a numerical algorithm for finding a minimum.

As one might suspect, it is possible that inequality constraints can sometimes be posed such that it is impossible to satisfy all of them. For example, the constraints $x >= 5$ and $x < 5$ are incompatible; only one of them can be true. Optimization problems where it is not possible to satisfy all constraints (whether equality or inequality constraints) are called *infeasible.*

The method of Lagrange multipliers can be applied for solving optimization problems with inequality constraints as well. To do so, we define new functions $\tilde{h}_j(\mathbf{x})$ as:

$$\tilde{h}_j(\mathbf{x}, s_j) = g_j(\mathbf{x}) + s_j^2, \text{ for each } j = 1, 2, \ldots, \ell.$$

The scalars s_j (yet to be determined) are called *slack variables.* The purpose of introducing these slack variables is to convert the inequality constraints $g_j(\mathbf{x}) \leqslant 0$ into *equality* constraints $\tilde{h}_j(\mathbf{x}, s_j) = 0$. Similar to the Lagrange multipliers introduced before, the values of the slack variables are determined by evaluating necessary conditions of the Lagrangian function.

It is easy to see that for each $j = 1, 2, \ldots, \ell$, the slack variable s_j indicate the "margin" by which the inequality constraint g_j is satisfied, namely, how far negative is the value of g_j at the candidate minimizer. In other words, if $s_j = 0$, then the inequality constraint $g_j(\mathbf{x}) \leqslant 0$ is actually satisfied as an equality, i.e., $g_j(\mathbf{x}) = 0$.

Informally, consider the example of finding the speed profile for traveling around a racing circuit in minimum time (this is actually an optimal *control* problem, but the concepts are similar). Here, an inequality constraint would be that the lateral force acting on the car be less than or equal to the maximum lateral friction force provided by the tires, for otherwise the car would "spin out." Intuitively, we expect that, to travel the circuit in minimum time, this inequality constraint will be met as an equality (i.e., the tires would be pushed to the limit or otherwise the driver would be "too careful / conservative"). This intuition can in fact be shown to be mathematically correct!

Returning to the solution of the inequality-constrained problem via the method of Lagrange multipliers, we formulate the Lagrangian function as:

$$L(\mathbf{x}, \mathbf{p}, \mathbf{s}) = f(\mathbf{x}) + \mathbf{p}^{\mathrm{T}} \begin{bmatrix} h(\mathbf{x}) \\ \tilde{h}(\mathbf{x}, \mathbf{s}) \end{bmatrix} \tag{1.10}$$

Note that the Lagrange multiplier $\mathbf{p}$ in this case is a vector of dimension $m+\ell$, and $\mathbf{s} = (s_1, \dots, s_\ell)$. As before, an unconstrained minimum of L is a minimum of f under the given constraints. We apply the necessary conditions to find the unconstrained minimum of L:

$$\begin{aligned}
\frac{\partial L}{\partial x_i}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{s}^*) &= \frac{\partial f}{\partial x_i}(\mathbf{x}^*) + \sum_{j=1}^{m} p_j^* \frac{\partial h_j}{\partial x_j}(\mathbf{x}^*) + \sum_{j=1}^{\ell} p_{m+j}^* \frac{\partial \tilde{h}_j}{\partial x_j}(\mathbf{x}^*, \mathbf{s}^*) = 0, && \text{for each } i = 1, \dots, n,\\
\frac{\partial L}{\partial p_i}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{s}^*) &= h_i(\mathbf{x}^*) = 0, && \text{for each } i = 1, \dots, m,\\
\frac{\partial L}{\partial p_{m+i}}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{s}^*) &= \tilde{h}_i(\mathbf{x}^*, \mathbf{s}^*) = 0, && \text{for each } i = 1, \dots, \ell,\\
\frac{\partial L}{\partial s_i}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{s}^*) &= 2p_{m+i}^* s_i^* = 0, && \text{for each } i = 1, \dots, \ell.
\end{aligned}$$

An additional necessary condition that arises in this case is that the Lagrange multipliers $p_{m+1}^*, p_{m+2}^*, \dots, p_{m+\ell}^*$ must be all nonnegative. These necessary conditions stated here for the inequality-constrained minimization problem are called the *Kuhn-Tucker (KT) conditions.*

Example 1.5. Minimize $f(x_1, x_2) = x_1^2 + x_2^2$ subject to the constraints

$$x_1 + 2x_2 \leqslant 15, \qquad 1 \leqslant x_1 \leqslant 10, \qquad 1 \leqslant x_2 \leqslant 10.$$

In this problem, we have $n = 2$, $m = 0$ (no equality constraints) and $\ell = 5$ (each of the last two inequalities contains two constraints: lower and upper bounds on the variables involved). Figure 1.2 illustrates the effect of these inequality constraints on the domain. In general, equality and inequality constraints reduce the domain of optimization.

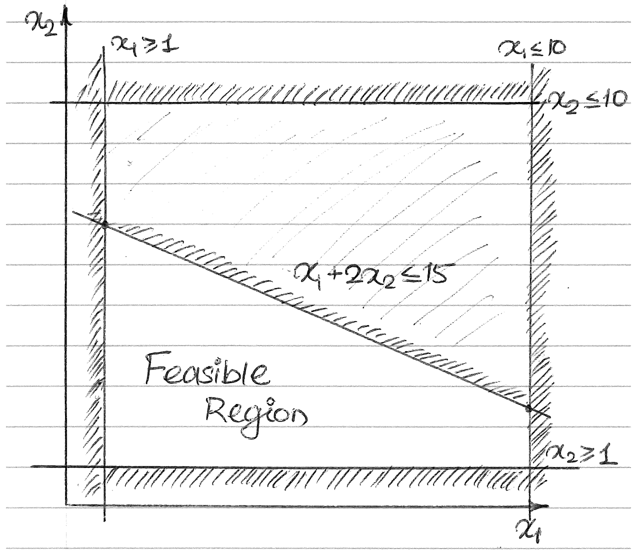


Figure 1.2: Illustration of inequality-constrained domain of optimization.

First, we rewrite the given inequality constraints in standard form:

$$g_1(\mathbf{x}) = x_1 + 2x_2 - 15 \leqslant 0, \qquad g_2(\mathbf{x}) = x_1 - 10 \leqslant 0, \qquad g_3(\mathbf{x}) = 1 - x_1 \leqslant 0,$$

$$g_4(\mathbf{x}) = x_2 - 10 \leqslant 0, \qquad g_5(\mathbf{x}) = 1 - x_2 \leqslant 0.$$

We define the Lagrangian function as in (1.10), and then write down the KT conditions as follows:

$$\begin{aligned}
\frac{\partial L}{\partial x_1} &= 2x_1^* + p_1^* + p_2^* - p_3^* = 0, & \frac{\partial L}{\partial x_2} &= 2x_2^* + 2p_2^* + p_4^* - p_5^* = 0,\\
x_1^* + 2x_2^* - 15 + (s_1^*)^2 &= 0, & x_1^* - 10 + (s_2^*)^2 &= 0,\\
1 - x_1^* + (s_3^*)^2 &= 0, & x_2^* - 10 + (s_4^*)^2 &= 0,\\
1 - x_2^* + (s_5^*)^2 &= 0, & p_i^* s_i^* &= 0 \text{ for each } i = 1, \ldots, 5.
\end{aligned}$$

These are 12 equations in 12 unknowns, namely, $x_1^*, x_2^*, p_1^*, \ldots, p_5^*, s_1^*, \ldots, s_5^*$. We can solve these equations to arrive at the following candidate minimum:

$$\mathbf{x}^* = (1, 1), \qquad \mathbf{p}^* = (0, 0, 2, 0, 2), \qquad \mathbf{s}^* = (2\sqrt{3}, \pm 3, 0, \pm 3, 0).$$

At this minimum, the fact that $p_1^* = p_2^* = p_4^* = 0$ indicates that the first, second, and fourth constraints are inactive. The fact that $s_3^* = s_5^* = 0$ indicates that the third and fifth constraints are active, i.e. satisfied as equalities. These observations are also clear from Fig. 1.2: $\mathbf{x}^* = (1, 1)$ lies at the boundary of the third and fifth constraints, but does not lie on the boundaries of the other three constraints. $\square$

1.4 Special Cases and Numerical Techniques

Constrained optimization problems are difficult to solve, in general. The primary sources of difficulty include the size of the domain, i.e. a high value of n, and the nature of the cost and constraint functions. Often, easy analytical expressions are not available for these functions, and the values of the cost and constraints at different points in the domain may be known only via a computer program such as a simulation of physical phenomena. Even if analytical expressions of the cost and constraint functions are available, they may be not simple enough to allow analytical solutions (such as the previous example) of the optimization problem.

The following special cases are significant because analytical are often possible, or else numerical techniques are particularly easy to apply:

Case 1 The cost and constraint functions are *linear* in the decision variables. In this case, the optimization problem is called a *linear programming problem,* and an efficient algorithm called the *simplex algorithm* can be used to solve it.

Case 2 The cost and constraint functions are *convex* in the decision variables. Informally, a convex function is “bowl-shaped,” e.g., a quadratic function. Convex programming problems can be solved by highly efficient numerical algorithms, and the *global minimum* can be found.

In all other situations, one must resort to numerical techniques, such as gradient descent and Newton’s method for unconstrained problems and Sequential Quadratic Programming (SQP) for constrained problems. The MATLAB® command `fmincon` encapsulates a set of such numerical techniques.

Chapter 2

Optimal Decision Sequences

The problem of optimizing a *sequence* of decisions is a part of optimal control, and applies to cases where the process to be controlled is best modeled as a sequence of discrete transitions. Graphs and difference equations are often used to mathematically model such processes.

2.1 Graphical Models of Processes

A graph $\mathcal{G} = (V, E)$ is a pair of two sets: a set V of *vertices* and a set E of *edges.* We will restrict this discussion to the case where both V and E have a finite number of elements. Each vertex in V can be uniquely associated with a state of the process to be modeled. Each edge in E is a pair of vertices, i.e., each edge is associated with the *transition* between two vertices. In other words, edges model control actions that change the state of the process.

Example 2.1 (**Traveling Salesman Problem).** There are 4 cities with known inter-city distances. For example: A traveling salesman is given the job of driving to each of these cities, starting from

	Boston, MA	Providence, RI	New York, NY	Albany, NY
Boston, MA	–	51	217	169
Providence, RI	51	–	182	163
New York, NY	217	182	–	151
Albany, NY	169	151	163	–

Table 2.1: Inter-city distances in driving miles.

Boston, with the restriction that he can visit each city exactly once. The optimal control problem of interest is to find the traveling salesman's trip that minimizes the total number of driving miles.

This problem can be modeled as graph $\mathcal{G} = (V, E)$. To this end, define the *state* (different from geographic state!) of the salesman's travels as the list of cities already visited. For example, the state BPN means that the salesman has visited Boston, Providence, and New York, in that order.

We can associate vertices in V with states, i.e.:

$$V = \{B, BP, BN, BA, BPN, BPA, \ldots\}.$$

Edges in the set E are associated with traveling from one city to another. For example, the edge (B, P) is associated with travel from Boston to Providence.

$$E = \{(B, N); (B, P); (B, A); (N, B); (N, A); (N, P); \ldots\}.$$

Edges are associated with state transitions, i.e. two vertices are linked by an edge. For example:

$$\text{State } BN \xrightarrow{\text{Edge } (N,P)} \text{State } BNP.$$

We can illustrate the graph $\mathcal{G}$ with a diagram indicating all vertices and all edges (see Fig. **??**).

Example 2.2 (Tower of Hanoi Problem). There are three pegs and several disks stacked on the first peg (see Fig. 2.1). Each disk is of a different diameter, and these disks are stacked such that for any given disk, the disks below it are of larger diameter. The optimal control problem of interest is to move the disks from the first peg to the third peg in a minimum number of moves, but with the following restrictions: only one disk can be moved at a time from one peg to another peg, and a disk can never be placed on top of a disk with smaller diameter.

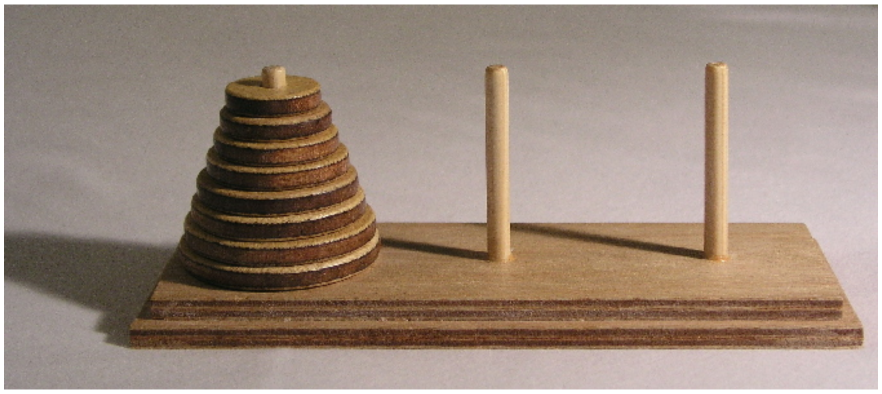


Figure 2.1: Illustration of the Tower of Hanoi problem.

This problem can be modeled as a graph $\mathcal{G} = (V, E)$. To this end, define the *state* as the locations of each disk. For simplicity consider the case with three disks, which we can label "small" (S), "medium" (M) and "large" (L). Each tower has three slots: "bottom", "middle" and "top," which we denote by T_{1T}, T_{2T}, and so on. We can list all possible states, some of which are shown in Table 2.2 where each row is a possible state. Each state can then be assigned to a unique vertex in V.

Edges in E are actions of the form

Move disk ________ from slot ________ to slot ________ .

	T_{1T}	T_{1M}	T_{1B}	T_{2T}	T_{2M}	T_{2B}	T_{3T}	T_{3M}	T_{3B}
v^1	L	M	S	–	–	–	–	–	–
v^2	–	–	–	L	M	T	–	–	–
v^3	–	–	–	–	–	–	L	M	T
v^4	L	M	–	S	–	–	–	–	–
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2.2: Description of states in the towers of Hanoi problem.

For example, define

$$e^1 \equiv \text{Move disk } \underline{\quad S \quad} \text{ from slot } \underline{\quad T_{1T} \quad} \text{ to slot } \underline{\quad T_{2B} \quad}.$$

This edge is associated with the transition from vertex v^1 to vertex v^4 in Table 2.2, i.e., $e^1 = (v^1, v^4)$, which causes the transition $v^1 \xrightarrow{e^1} v^4$.

Example 2.3 (**Path-planning Problem).** Recall the problem of planning the path of an aircraft to minimize exposure to threat. It is possible to approximate this problem by discretization. Specifically, we formulate a grid of points in the environment (see Fig. 2.2), and convert the path-planning problem to a problem of finding a sequence of grid points.

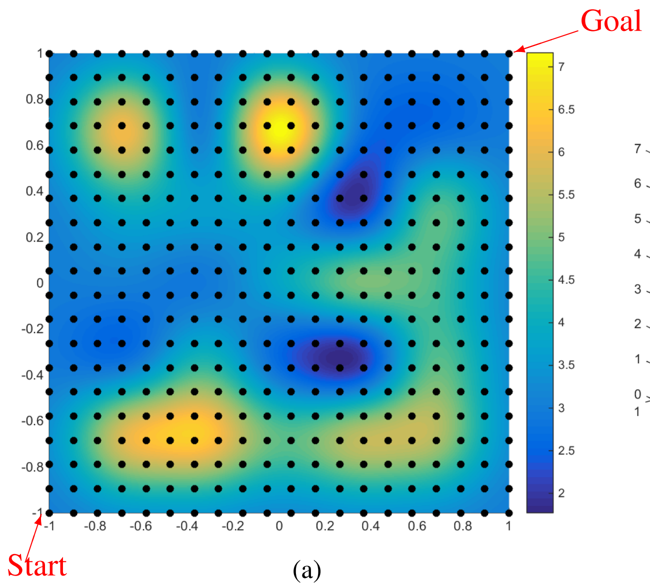
This problem can be modeled as a graph $\mathcal{G} = (V, E)$. Here, each vertex in V is associated with a grid point, and each edge is associated with a pair of adjacent vertices. The pictorial illustration of this graph is easy: the vertices can be drawn in a grid, as shown in Fig. 2.2(a), and edges can be drawn as connecting lines between adjacent grid points.

The coordinates in a prespecified Cartesian coordinate axis system of the i^{th} grid point are denoted by x^i. We consider a strictly positive scalar field $c : \mathcal{W} \to \mathbb{R}_{>0}$, called the *threat field*, which represents unfavorable regions with higher intensity. Each vertex in V is uniquely associated with a grid point, and labeled with superscripts as $v^1, v^2, \ldots$

To use graphical models of processes for finding optimal sequences, we should be able to evaluate the cost of any given sequence of decisions. To this end, an *edge transition cost function* $\bar{g} : E \to \mathbb{R}_{\geqslant 0}$ is introduced. This function assigns to every edge in the set E a nonnegative scalar value called the *transition cost* of that edge.

Example 2.1 (**Traveling Salesman Problem).** In this problem, we are interested in minimizing the total driving distance. Therefore, an appropriate edge transition cost function is the function that assigns to each edge the driving distance between the two cities associated with the edge. For example:

$$\bar{g}((B, N)) = 217, \qquad \bar{g}((B, P)) = 51, \qquad \bar{g}((N, A)) = 151, \qquad \text{and so on.}$$



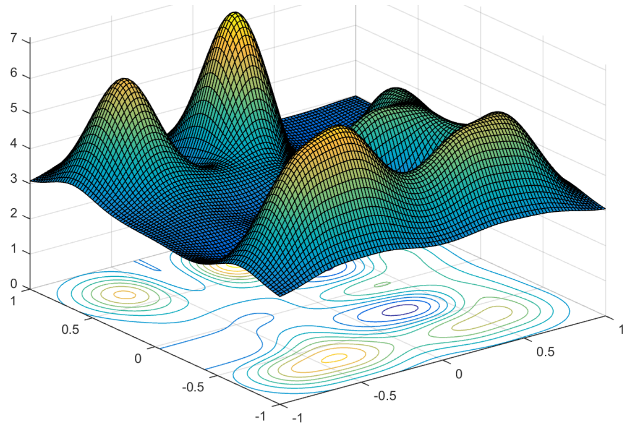


Figure 2.2: Threat intensity map with grid overlaid.

Example 2.2 (Tower of Hanoi Problem). In this problem, we are interested in minimizing the total number of moves. Therefore, an appropriate edge transition cost function is simply the function that assigns a unit cost to each edge, i.e.,

$$\bar{g}(e) = 1, \qquad \text{for every } e \in E.$$

Example 2.3 (Path-planning Problem). In this problem, we are interested in minimizing the total exposure to threat. Therefore, an appropriate edge transition cost function is the average of the threat exposure at each of the two grid points associated with the edge:

$$\bar{g}((v^i, v^j)) = (c(\mathsf{x}^i) + c(\mathsf{x}^j))/2. \tag{2.1}$$

$$\tag{2.2}$$

A *path* in the graph $\mathcal{G}$ between two given vertices v^{i_s} and v^{i_g} is a sequence $\mathbf{v} = (v_0, v_1, \ldots, v_P)$, without repetition, of successively adjacent vertices with $v_0 = v^{i_s}$ and $v_P = v^{i_g}$. Note that indices of vertices in this sequence are denoted using *sub*scripts. The *cost* $\mathcal{J}(\mathbf{v}) \in \mathbb{R}_{\geqslant 0}$ of this path is the sum of edge transition costs, i.e.,

$$\mathcal{J}(\mathbf{v}) = \sum_{k=1}^{P} \bar{g}((v_{k-1}, v_k)).$$

We can formulate the problem of optimal control of processes modeled by graphs as a problem of finding a path in a graph with minimum cost. To this end, it is necessary to identify one or more initial states of the process, and one or more desired (goal) states, which can then be provided as the vertices v^{i_s} and v^{i_g} for the path optimization problem.

Example 2.1 (Traveling Salesman Problem). In this problem, the initial state is that only one city, namely Boston, is visited, i.e., the vertex in V associated with the state B is v^{i_s}. The goal state is that all four cities be visited, in any order. Therefore, each of the following vertices is acceptable as v^{i_g} : $BNPA, BNAP, BAPN, BANP, BPAN, BPNA$.

Example 2.2 (Tower of Hanoi Problem). In this problem, the initial state is of having all three disks on the first tower, and the goal state is of all three disks on the third tower. These states are associated with vertices v^1 and v^3, respectively, in Table 2.2, i.e., $i_\mathrm{s} = 1$ and $i_\mathrm{g} = 3$.

Example 2.3 (Path-planning Problem). The initial and goal vertices are shown in Fig. 2.2.

2.1.1 Label-correcting Algorithms

The standard shortest path problem can be solved by a class of algorithms called the *label-correcting algorithms*. Well-known examples of label correcting algorithms include the Bellman-Ford, the Dijkstra, and the A* algorithms.

A label-correcting algorithm progressively searches for the least cost path starting from v^{i_s} and ending at vertex $v^{i_\mathrm{g}} \in V$, by iteratively reducing an estimate of the least cost to the initial vertex v^{i_g}, called the *label* of the vertex v^{i_g}. To do so, the algorithm maintains the label of all vertices in V. Let $d : V \to \mathbb{R}_+$ denote the label function, i.e., $d(v^i)$ is the current estimate of the least cost from v^{i_s} to v^i. The algorithm also maintains a set $\mathcal{P}$ of vertices, called the *fringe*, that contains the vertices whose labels can potentially be reduced from their current value (sometimes referred to as the OPEN list), and it maintains a *backpointer* function $b : V \to V$, which records the immediate predecessor of each node $v^i \in V$ in the optimal path from v^{i_s} to v^i. Finally, the algorithm maintains a function called the *marker*, which assigns every vertex in V as either "visited" (V), "unvisited" (U), or "closed" (C). The pseudo-code for a general label-correcting algorithm is shown in Fig. 2.3.

Here, the procedures REMOVE and INSERT are to be read as "Remove from $\mathcal{P}$" and "Insert into $\mathcal{P}$," respectively. These procedures can be implemented in different ways. In Dijkstra's algorithm, which is a widely used label-correcting algorithm, the REMOVE procedure removes the vertex in $\mathcal{P}$ with minimum label. The INSERT procedure is often implemented using a binary tree or a Fibonacci heap, which help to efficiently maintain $\mathcal{P}$ as a list of vertices sorted according to their label values. For Dijkstra's algorithm, the $StopCondition$ in Line 12 is evaluated as follows:

$$StopCondition := \begin{cases} True & \text{if } v^i = v^{i_\mathrm{g}} \text{ or if } \mathcal{P} \text{ is empty}, \\ False & \text{otherwise.} \end{cases}$$

The following properties can be proven about Dijkstra's algorithm:

- The algorithm terminates after a finite number of iterations.
- After the termination of the algorithm, the label of the goal vertex is equal to the minimum cost of a path from the start vertex.
- During the execution of the algorithm, a vertex once removed from the fringe $\mathcal{P}$ is never inserted back into the fringe.

– Note: This property does not have to explicitly enforced, i.e., it automatically holds true when the algorithm is implemented as shown in Fig. 2.3.

General Label-Correcting Algorithm

Input: $v^{i_s}, v^{i_g} \in V$ **Output:** Label d, Backpointer b

```
INITIALIZE($v^{i_s}$)
 1: $\mathcal{P} := \{v^{i_s}\}$, $d(v^{i_s}) := 0$, Marker($v^{i_s}$) := $V$
 2: for all $v^j \in V \backslash \{v^{i_s}\}$ do
 3:     $d(v^j) := \infty$, $b(v^j) := \varnothing$, Marker($v^j$) := $U$
 4: end for
 5: $StopCondition = False$

MAIN
 1: INITIALIZE($v^{i_s}$)
 2: while ($StopCondition = False$) do
 3:     $v^i :=$ REMOVE($\mathcal{P}$)
 4:     Marker($v^i$) := $C$
 5:     for all $v^j \in V$ such that $(v^i, v^j) \in E$ and Marker($v^j$) $\neq C$ do
 6:         if $d(v^i) + g((v^i, v^j)) < d(v^j)$ then
 7:             $d(v^j) := d(v^i) + g((v^i, v^j))$
 8:             $b(v^j) := v^i$, Marker($v^j$) := $V$
 9:             $\mathcal{P} :=$ INSERT($\mathcal{P}, v^j$)
10:         end if
11:     end for
12:     Evaluate $StopCondition$
13: end while
```

Figure 2.3: Pseudo-code for the general label-correcting algorithm.

Example 2.4 (Path-planning Problem). We illustrate the execution of Dijkstra's algorithm on a small grid on 9 vertices as shown in Fig. 2.4. The values of the threat field at each of these grid points are:

	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9
$c(\mathsf{x})$	3.067	3.306	3.051	3.014	3.821	3.226	3.081	3.661	2.9861

Therefore, the edge transition cost functions according to Eqn. (2.1) are as follows:

	$(1,2)$	$(2,3)$	$(4,5)$	$(5,6)$	$(7,8)$	$(8,9)$	$(1,4)$	$(4,7)$	$(2,5)$	$(5,8)$	$(3,6)$	$(6,9)$
$g((v^i, v^j))$	3.186	3.178	3.417	3.523	3.371	3.323	3.041	3.048	3.653	3.741	3.138	3.106

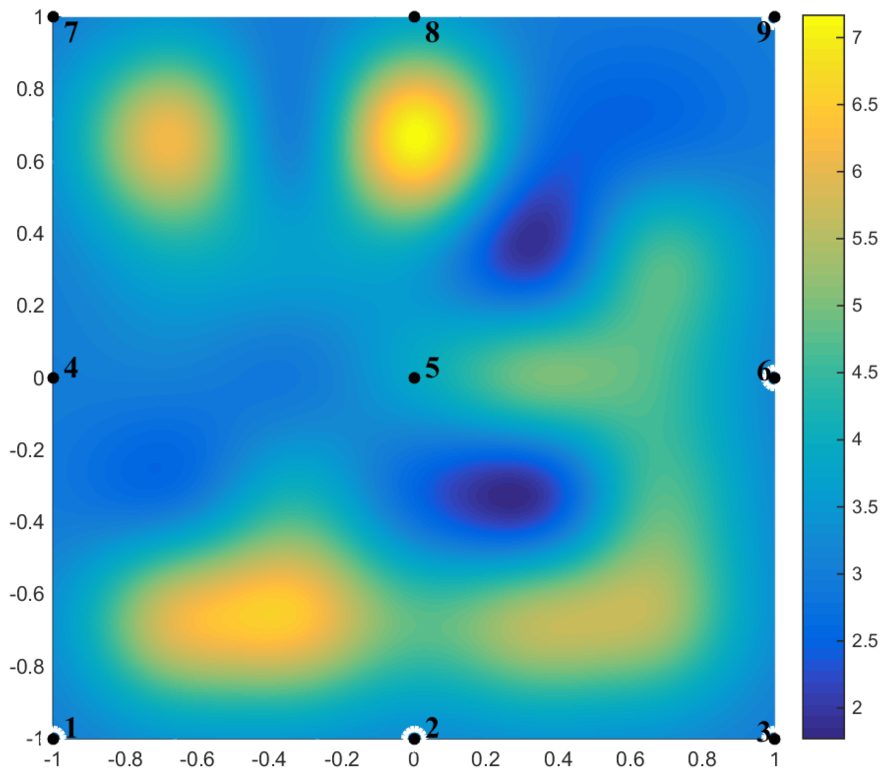


Figure 2.4: Threat intensity map with 9-point grid overlaid.

The start and goal vertices are as indicated, i.e., $i_\mathrm{s} = 1$, and $i_\mathrm{g} = 9$. To understand the execution of Dijkstra's algorithm for this example, we track the changes made by the algorithm to the label d, backpointer b, marker M, and the fringe $\mathcal{P}$. The initialization step (Line 1 of MAIN) results in the following values for these quantities:

	v^1	v^2	v^3	v^4	v^5	v^6	v^7	v^8	v^9
$d(v)$	0	∞	∞	∞	∞	∞	∞	∞	∞
$b(v)$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
$M(v)$	V	U	U	U	U	U	U	U	U
$\mathcal{P}$					$\{v^1\}$				

At Line 3 of MAIN, the REMOVE procedure returns $v^i = v^1$ because v^1 is the sole element of $\mathcal{P}$. Next, in Line 4, the algorithm assigns $M(v^1) := C$. For Line 5, the vertices v^j that satisfy the stated description are v^2 and v^4. Examine the execution of Lines 6–11 for $v^j = v^2$. Here, the condition in Line 6 is trivially satisfied because $d(v^2) = \infty$. Therefore, Lines 7–9 are executed.

Lines 7–9 are then repeated with $v^j = v^4$, and the new values of $d, b, m,$ and $\mathcal{P}$ are assigned as:

	v^1	v^2	v^3	v^4	v^5	v^6	v^7	v^8	v^9
$d(v)$	0	3.186	∞	3.041	∞	∞	∞	∞	∞
$b(v)$	$\varnothing$	1	$\varnothing$	1	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
$M(v)$	C	V	U	V	U	U	U	U	U
$\mathcal{P}$	$\{v^2, v^4\}$								

Next, the algorithm loops back to Line 3, where the REMOVE procedure returns v^4 because it has the minimum label among the vertices currently within $\mathcal{P}$ (which are v^2 and v^4). For Line 5, the vertices v^j that satisfy the stated description are v^5 and v^7. Note that, although $(v^1, v^4) \in E$, the vertex v^1 does not fit this description because of its marker value, namely $M(v^1) = C$. Lines 6–10 are then executed for each of v^5 and v^7, and the new values of $d, b, m,$ and $\mathcal{P}$ are assigned as:

	v^1	v^2	v^3	v^4	v^5	v^6	v^7	v^8	v^9
$d(v)$	0	3.186	∞	3.041	6.458	∞	6.089	∞	∞
$b(v)$	$\varnothing$	1	$\varnothing$	1	4	$\varnothing$	4	$\varnothing$	$\varnothing$
$M(v)$	C	V	U	C	V	U	V	U	U
$\mathcal{P}$	$\{v^2, v^5, v^7\}$								

The minimum label among vertices in $\mathcal{P}$ is v^2 because $d(v^2) < d(v^5) < d(v^7)$, and therefore in the next iteration, the REMOVE procedure returns $v^i = v^2$, and the algorithm continues. The algorithm terminates after a few iterations, and the following values of d and b (the marker and fringe are not important outputs after the termination of the algorithms).

	v^1	v^2	v^3	v^4	v^5	v^6	v^7	v^8	v^9
$d(v)$	0	3.186	6.364	3.041	6.458	9.503	6.089	9.459	12.61
$b(v)$	$\varnothing$	1	2	1	4	3	4	7	6

The goal vertex is v^9, and therefore $d(v^9) = 12.61$ is the minimum cost of a path from $v^{i_s} = v^1$ to the goal vertex. To find this path, we trace the backpointers of vertices backwards from the goal vertex until we reach the start vertex, i.e., the last vertex in the optimal path is v^9, the second-to-last is $b(v^9) = v^6$, the third-to-last is $b(v^6) = v^3$ and so on. Continuing backwards this way until the start vertex, we find that the optimal path is $\mathbf{v}^* = (v^1, v^2, v^3, v^6, v^9)$.