University of Sussex

Resource Allocation and Markovian Processes Under Dynamic Programming

Ugur Eren Canakci*

School of Mathematical and Physical Sciences, University of Sussex

Abstract

This Master's thesis is the study of dynamic programming and its applications to real world problems from the book "Applied Dynamic Programming" By Richard Bellman and Stuart E. Dreyfus, published in 1962. The questions this thesis is seeking to answer are "is there a strategy that guarantees the optimization of processes that require an investment or resource allocation, and if so, how is the theory applied to real world problems?", "how can we maximize the averaged outcomes of non-deterministic processes with the strategies we have?" and "what can we do when analysis techniques are not enough in the investigation of optimal outcomes, especially in discrete settings or functions with uncertainties?". The answer for the first question comes in the form of one and two dimensional allocation processes, where the partition of resources, that are either discrete or continuous, are investigated, and optimized over the space of all possible allocations. The second question is studied under the Markovian mindset, where at each stage of the process, a strategy is invoked to dictate the transitions over the state space, where the best long term strategies are obtained with respect to the outcomes of the processes. With various techniques such as Lagrange multipliers and policy improvement routines introduced into the problem, the optimal controls are obtained with ease. This thesis is contributes to the literature with one correction on the results, confirmation in others, and adoration on the elegance of the mindset of dynamic programming.

*Corresponding author: uc38@sussex.ac.uk

Contents

List of Figures

Introduction

The aim of this thesis is answering the three questions in the abstract by introducing five optimization problems from industry and business settings, and solving the problems with the dynamic programming(DP for short) principles introduced before the description of each. All questions and underlying theory is from [1], and the thesis additionally presents numerical results obtained in MATLAB with figures and tables, as well as the codes for each problem in the appendix.

The problems to be studied are processes that present outcomes from a state space S with respect to the reactions, or in other words, controls, given to each outcome. We denote the outcomes of a problem at each step $n \in \mathbb{N}$ by x_n, and looking at x_n we choose a control $y_n = y_n(x_n)$ from a set of applicable controls Y_n, and the problem presents us with another outcome x_{n+1}. This translates to the equation

$$x_{n+1} = r(x_n, y_n) \tag{1}$$

for some function r. Our focus is to find the control vectors $(y_1, y_2, \ldots, y_N)$ that optimizes the total output of the process. Depending on the questions, we will prefer either maximizing or minimizing this output, and for that reason, we work over the returns of control vectors, which are elements of the policy space $\Pi_{1\leq n\leq N} Y_n$, with various dynamic programming techniques.

The processes to be investigated in this thesis have real outputs that are incurred with specific controls or strategies we decide to apply. For any random multistage process, there can be introduced a basic formula[1, page ix]

$$f(p) = \max_q \left[H(p, q, f(T(p, q)))\right] \tag{2}$$

where p is the "state vector" of the initial stage of the process, q is the control vector applicable to the initial state p, $T(p, q)$ is the outcome of giving the reaction q to the state p of the process, and H is an arbitrary (but useful) functional, defined on all stages and states of the process.

Recognize that $f(T(p, q))$ is present in the definition of $f(p)$. Keeping in mind that we "initiate" the process on state p, and by giving the best reaction q^*, we obtain the next outcome $T(p, q^*)$ from the process. With this we obtained a "sub-process" of our main process, not including the initial stage(whose outcome was p). This sub-process has the initial state $T(p, q^*)$, and we optimize the subprocess as well as the initial stage by choosing one of the optimal control vectors. This structure of f outlies the logic under what is called "the principle of optimality", which will be explained in the section "Multidimensional Allocation Processes".

While working on the problems, calculus may be used depending on the nature of the problems, yet it is not definitive in obtaining the best results. Firstly, whenever introduced,

the functions related to the problems are either given to be "smooth", i.e. fitting some differentiability assumptions, or approximations to the original problems are constructed, which are likely to overfit whatever prior information is used in the construction. Second of all, if we are working on discrete spaces, most calculus techniques bear no significant help for the problem due to the nature of the spaces. Dynamic programming offers a solution to these problems in an intuitive and clear to understand way.

The study of dynamic programming had an increase of interest in magnitudes around the times World War II due to expected reasons. With the search on new techniques, a need for well defined theory emerged, and we started seeing Bellman's works such as [2], [3], [4], and [1]. Bellman's work expanded onto almost all disciplines of science, supplying a mathematical foundation to problems of sequential nature.

There are three main sections in this paper. Every section is focused on one type of problem and some examples of problems solvable by the strategies and algorithms, as well as the outcomes and plots of various results are given.

The first chapter is focused on one dimensional allocation processes, where a single type of resource is partitioned over multiple stages of a process in order to obtain an outcome. As a real world example, a cargo problem (that is discrete in allocations) is investigated, where various different types of goods are constrained by a certain constraint on total chosen to put in a cargo vehicle so that the return of one cargo transportation is maximized.

The second chapter extends the allocation processes to multiple different resources. In doing so, the complexity of the calculations increase in multitudes, so the technique of Lagrange multipliers is introduced. For the examplary questions, a new cargo problem is introduced with a non-deterministic setting at base, as well as a question that focuses on the management of two types of investments to various departments of a company, where the calculations are simplified with the usage of Lagrange multiplier.

The final chapter focuses on Markovian decision processes, where control elements introduce a transition probability and return distribution over the state space of the processes. The first question that is solved is a taxicab problem, where the driver is supposed to maximize the average gains per carriage by choosing the optimal strategies on each state. The second question that is solved is a tire manufacturing problem where the cost of operating a machine is minimized by the right choices of production or repairment on each specific state of the machine.

Below is a short list of suggestions for further studies of resource allocation and Markovian decision problems:

1. For a deeper delve into the theoretical underpinnings of resource allocation problems presented in this thesis, [5] is a great resource. On type of problems involving the supply-

demand scheme of various types of goods (similar to the flyaway kit problem explained in the section "multidimensional allocation processes"), a generalized setup for the optimal cost of the whole process is derived, reaching the same outcomes with the calculus of variation technique.

2. If one is interested in a more abstract investigation of the dynamic programming principles, [6] explains the outcomes of monotonicity and discount/contraction assumptions on the processes, with the help of functional analysis tools at bay. [7] and [8] are similar collection to [6] in the sense of depth and scope, with a focus on Markovian decision processes.
3. For solutions of allocation problems with vast amounts of control variables, either due to having large number of stages in the processes, or due to controlling many different resources at each stage, one could take a look at [9], where the authors suggests introducing "basis" functions defined to produce results explanatory for the outcomes of the process (page 2991). [10] is another example for allocation problems industrial settings, similar to tire manufacturing problem, yet focused on an integer allocation scheme similar to the cargo problems presented.
4. [4] presents a gambling problem where various amounts of bets are allocated to $N \in \mathbb{N}$ signals. The condition that brings the indeterministic nature into the problem is the "noise"; for each signal i there is assumed a probability distribution over all signals $1, \ldots, N$ such that when signal i is activated, some other signal j will have a likelihood of "appearing as activated", i.e. creating noise. The gambler aims to find how their fixed budget $\mathbf{x}$ should be partitioned on each signal so to obtain maximum return

$$\sum_{i=1}^{N} r_i x_i \mathbb{1}_i \tag{3}$$

where x_i is the bet/investment on signal i, r_i is the reward/return coefficient of signal i, and $\mathbb{1}_i$ is the activation indicator for signal i;it equals 1 if i is not noise, and 0 otherwise.

One Dimensional Allocation Process [1]

We start our discussion of dynamic programming by focusing on one dimensional resource allocation problems. Assume that we are working on a process that is defined on only one type of resource, i.e. each stage has only one independent variable affecting the outcome of that stage. There will be assumed a non-negative amount of resource $\mathbf{x}$ for the processes of this nature, where $\mathbf{x}$ will be partitioned into the stages of the problem as $(x_1, x_2, \ldots, x_N)$. Outcomes of each stage $n = 1, \ldots, N$ is represented by the function $g_n = g_n(x_n)$.

For a one dimensional allocation process with N stages, we would like to obtain the maximum of

$$R(x_1, x_2, \ldots, x_N) = g_1(x_1) + g_2(x_2) + \ldots + g_N(x_N) \tag{4}$$

where R is the total return we will obtain out of that process, decided by the partition $(x_1, x_2, \ldots, x_N)$. Recall that the partition of a number requires the parts to sum up to the original number, as well as no parts being less than zero:

$$x_i \geq 0, i = 1, 2, \ldots, N, \tag{5}$$

$$x_1 + x_2 + \ldots + x_N \leq x. \tag{6}$$

Our main goal is to optimize the returns, i.e. reach a maximum (or minimum, respectively) outcome from the return function R by "imbedding the return within a family of allocation processes"[1, page 13]. Instead of trying to find various partitions of $\mathbf{x}$, we allocate a certain x_N to the stage N first, then allocate x_{N-1} to the stage $N-1$, and so on. Since the allocation is done stage by stage, the allocation process has become dynamic in nature.

Let the sequence $\{f_n(\mathbf{x}) : n = 1, 2, \ldots, N\}$ denote the optimal returns from allocating the resource of amount $\mathbf{x}$ to stages from 1 to n under constraints 5 and 6. Then

$$f_n(\mathbf{x}) = \max_{(x_1, x_2, \ldots, x_n)} \{g_1(x_1) + g_2(x_2) + \cdots + g_n(x_n)\}. \tag{7}$$

When we are provided that $g_i(0) = 0, i = 1, 2, ..., N$, we see that $\mathbf{x}$ being equal to zero results in

$$R(x_1, ..., x_N) = R(0, 0, ..., 0) = \sum_{i=1}^{N} g_i(0) = \sum_{i=1}^{N} 0 = 0. \tag{8}$$

Also recognize that whenever we have only one stage in a process,

$$f_1(\mathbf{x}) = g_1(x_1), x_1 = \mathbf{x}. \tag{9}$$

The allocations onto an allocation process will always start from the final stage, whose return function is g_N. This creates a particularly important leeway when we want to analyse problems where the partition of resources is done on (possibly) infinitely many stages. If we were to

start an infinite allocation process from g_1, then the rest of the stages would still comprise an infinite stage process, leaving us with the result $\max_{(x_i)_{i=1}^{N}} \{g_1(x_1) + g_2(x_2) + \cdots + g_n(x_n) + R'\}$ for some $R' \in [0, \infty]$ when n allocations are made and $\mathbf{x} - (x_1 + \cdots + x_n)$ resource remains. When the other direction is followed, i.e. when we start the allocations from an arbitrary n'th activity and go towards g_1, we encounter a finite allocation process, whose result is dependent on n. Taking n to infinity on the results of finite problems solves the infinite allocation problem easily.

Recursive Definition of f

Assume for some arbitrary stage n and arbitrary amount of resources $\mathbf{x} \in \mathbb{R}^{\geq 0}$, an optimal return $f_n(\mathbf{x})$ is given. In order to calculate $f_{n+1}(\mathbf{x})$,we will allocate all possible x_{n+1}'s in the interval $[0, \mathbf{x}]$ to the $n+1$'th activity, and the rest of the resources $\mathbf{x} - x_{n+1}$ will in total be allocated to the rest of the stages $n, n-1, n-2, ..., 2, 1$. In short, with the addition of the $n+1$'th activity, the possible returns of all stages are comprised of

$$S_{n+1}(\mathbf{x}) := \{g_{n+1}(x_{n+1}) + f_n(\mathbf{x} - x_{n+1}) : x_{n+1} \in [0, \mathbf{x}]\} \tag{10}$$

over all possible initial resource allocations x_{n+1}. Realize that $f_n(x - x_{n+1})$, the maximum return from the previous n stages with remaining resource $\mathbf{x} - x_{n+1}$, occurs in 10, simply because any non-optimal allocation on the rest of the stages will result in less total return over $n+1$ stages. Assuming $S_{n+1}(\mathbf{x})$ contains a maximum value, we can choose x^*_{n+1} so that we obtain the maximum value in $S_{n+1}(\mathbf{x})$:

$$x^*_{n+1} \in \arg\left\{\max_{0 \leq x_{n+1} \leq \mathbf{x}} \{g_{n+1}(x_{n+1}) + f_n(\mathbf{x} - x_{n+1})\}\right\}, \tag{11}$$

and so obtain $f_{n+1}(\mathbf{x})$ by using x^*_{n+1}:

$$f_{n+1}(x) = g_{n+1}(x^*_{n+1}) + f_n(\mathbf{x} - x^*_{n+1}) = \max_{x_{n+1} \in [0,x]} \{g_{n+1}(x_{n+1}) + f_n(x - x_{n+1})\}. \tag{12}$$

Cargo Allocation Process

Suppose we have a cargo vehicle with which we carry various different types of goods. Each type of good has a specific weight and value per item associated with it. What's in our interest is the best choice of various types of goods to carry at each travel, to obtain the maximum return of operation.

If the goods are possible to divide into as small pieces as needed, then the solution is pretty simple; we simply select whichever type of good has the most value per weight, and fill up the vehicle to its maximum capacity. When indivisible goods are present, however, we can only take $0, 1, 2, \ldots$ of these goods.

Time Complexity of Direct Comparison Between Allocations One can conjure the very basic attempt of comparing all possible choices of goods, i.e.all possible allocations of the main resource, and compare the returns from each allocation to find which allocations provide the maximum return. Let's set up an example to see the time constraints that may come up with this . Assume we have n types of goods we can put into our cargo vehicle for some fixed $n \in \mathbb{N}$. We would like to see all the possible combinations of allocation of types, given that the sum of the weights of items is less than or equal to the maximum capacity of our vehicle. We are going to construct allocation vectors $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ where each allocation α_i takes integer values between 0 and a maximum amount

$$\beta_i = \lfloor \frac{\texttt{maximum\_capacity}}{w_i} \rfloor, \tag{13}$$

that represent how many items of each type has been selected to put in the vehicle. After filtering the allocations that result in higher total weight than the maximum capacity of the vehicle, we will compare the total profit from each vector that's viable for the maximum capacity.

Looking at the expected time to finish all calculations with direct comparison of outcomes between each possible allocation, we observe some remarks, listed below:

1. The allocations will be filtered with respect to the total resource allocation, right after each is created, creating no increase in the order of needed time to finish the calculation.
2. Total value calculation of each viable allocation $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ will be a simple vector product between the allocation and the vector that holds the values of each type of item. There will be less calculations than the number of all allocation vectors, including the ones that are filtered out . Again, there is no significant increase in the order of expected time to finish the calculation.

Then the major time consuming activity will be the creation of all possible allocation vectors. For each entry α_i, we will choose one number from 0 up to β_i, resulting in a multiplier $\binom{\beta_i+1}{1}$ for all $i = 1, 2, \ldots, n$ on the time required for calculations. Hence, we will have

$$\Pi_{i=1}^{n} \binom{\beta_i + 1}{1} = \Pi_{i=1}^{n} \ (\beta_i + 1) \tag{14}$$

Setting $\beta = \min_i\{\beta_i\} = \min_i\{\lfloor \frac{maximum\_capacity}{w_i} \rfloor\}$ we find a lower limit for the number of allocation vectors:

$$\Pi_{i=1}^{n} \ (\beta_i + 1) \geq \Pi_{i=1}^{n} \ (\beta + 1) = (\beta + 1)^{\,n}$$

Hence, a program running this algorithm will need time at least of order $\mathcal{O}\left((2)^n\right)$ since β can.

Results The weights and sizes for eight types of goods are presented in [1, page 31]:

```
weights = [20; 18; 14; 12; 10; 16; 22; 24];
values = [72; 60; 40; 27; 20; 50; 85; 96];
```

These vectors are read as such: the first entries in each vector is of the first type of item, second entries are for second type, and so on.

We obtain the optimal allocations and returns of each said allocation as below:

type of good:	1	2	3	4	5	6	7	8	Returns
weight capacity: 10	0	0	0	0	1	0	0	0	20
weight capacity: 20	1	0	0	0	0	0	0	0	72
weight capacity: 30	0	0	0	0	0	0	0	1	96
weight capacity: 40	0	0	0	0	0	1	0	1	146
weight capacity: 50	0	0	0	0	0	0	0	2	192
weight capacity: 60	0	0	1	0	0	0	1	1	221
weight capacity: 70	0	0	0	0	0	0	1	2	277
weight capacity: 80	0	0	0	0	1	0	1	2	297
weight capacity: 90	0	0	0	0	0	0	3	1	354
weight capacity: 100	0	0	0	0	0	0	0	4	384

Figure 1: *The optimal allocations of eight types of good for maximum return per each transport, for weight capacities from* 10 *to* 100

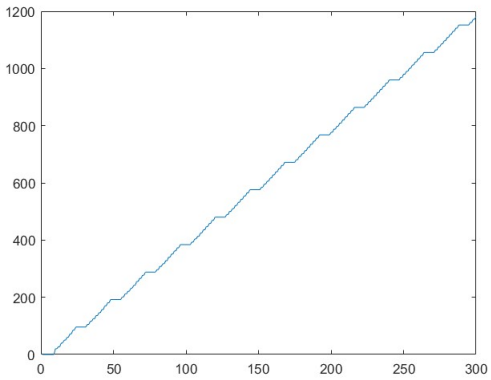

Figure 2: *The return due to the optimal allocation of types of goods for each capacity, from 1 to 300*

The code for the solutions is in the appendix.

Multidimensional Allocation Processes [1]

Allocation processes with a single type of resource was explained in the previous section. Now, we enbroad the study of allocation processes onto a wider class of problems, which can have finitely many types of resources $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^m$ to be allocated into each stage, instead of a single type of resource. Each stage $n = 1, 2, \ldots, N$ will produce an outcome $g_n(x_n^1, x_n^2, \ldots, x_n^m)$ due to the allocations $(x_n^1, x_n^2, \ldots, x_n^m)$. Hence, problems of this type can be utilized under the class of multidimensional allocation processes in the dynamic programming studies.

For simplicity, take a multidimensional allocation process with N stages and two types of resources, $\mathbf{x}$ and $\mathbf{y}$. Set $g_k(x_k, y_k)$ to be the return from the k'th stage of the process, $k \in 1, 2, \ldots, N$ due to allocating x_k and y_k with constraints

1.
$$x_k, y_k \geq 0,\ k = 1, 2, \ldots, N \tag{15}$$

2.
$$\sum_{k=1}^{N} x_k \leq \mathbf{x}, \tag{16}$$

3.
$$\sum_{k=1}^{N} y_k \leq \mathbf{y}. \tag{17}$$

Then the total return R from this process is obtained by creating an allocation vector $(x_1, x_2, \ldots, x_N; y_1, y_2, \ldots, y_N)$ with the constraints above, and adding up all the outcomes $g_k(x_k, y_k)$, $k = 1, 2, \ldots, N$:

$$R(x_1, x_2, \ldots, x_N; y_1, y_2, \ldots, y_N) = g_1(x_1, y_1) + g_2(x_2, y_2) + \cdots + g_N(x_N, y_N) \tag{18}$$

Extension of One Dimensional Allocation Processes

By the formulation above, we can extend the context of one dimensional allocation processes. For problems with a single type of resource, various constraints can be defined on quantities that are obtained from functions whose argument is the resource. Keeping the formulation same with the one dimensional case, we can set two functions α_k, β_k for each stage k of the process to represent the constraints on the resource $\mathbf{x}$. For any allocation vector $(x_1, x_2, \ldots, x_N)$, we can think of this vector as constrained by two quantities C and D such that

1. $\alpha_k(x_k), \beta_k(x_k) \geq 0,\ k = 1, 2, \ldots, N$

2. $\sum_{k=1}^{N} \alpha_k(x_k) \leq C$,
3. $\sum_{k=1}^{N} \beta_k(x_k) \leq D$.

We are again looking for possible allocations that result in either the maximum or minimum value R can obtain where R has the form of 4.

Time Complexity of Direct Comparison Between Allocations

The multidimensional allocation problems have their complexity increased in magnitudes compared to one dimensional problems, and to be precise, by the same magnitude for each dimension. One dimensional cases already need time of order $\mathcal{O}(2^N)$ (refer to the subsection in Section 1 with the same title) for the calculation of optimal allocation $(x_1^*, x_2^*, \ldots, x_N^*)$ obeying the constraints given before.

We can safely put 2 in the argument of $\mathcal{O}$ simply because there has to be at least two choices of resource allocation at each stage, and order $\mathcal{O}(2^N)$ is already scaling too fast per additional stage for the problems to be accessible.

For each dimension, i.e. for each type of resource, there will be another vector that has to be found in an N-dimensional finite space, in accordance with other vectors in resulting an outcome. Recall that a return or a cost occurs when an allocation is done, and so to obtain the total return R, we have to find every single allocatable value(s) for each type of resource and for each stage of the process. Hence, for a process with N stages that is incurred by M different types of resources will have an allocation vector (or a matrix, call it however you want) that has $N \cdot M$ entries. Hence, direct comparison of allocations for optimal returns from this process will require time of order $\mathcal{O}(2^{NM})$.

Principle of Optimality for Multidimensional Allocation Processes

The pillar of dynamic programming for allocation processes is the principle of optimality.In both one dimensional and multidimensional allocation processes, the sequential approach to problems are allowed by this principle.

Consider an allocation process of N stages with two types of resources, $\mathbf{x}$ and $\mathbf{y}$. Assume we would like to maximize

$$g_1(x_1, y_1) + g_2(x_2, y_2) + \cdots + g_N(x_N, y_N), \tag{19}$$

where each stage k of the process outputs a return $g_k(x_k, y_k)$ under the constraints 15, 16 and 17. The main strategy is the same with one dimensional case: seperate the initial state, whose return is given by g_N, from the rest of the process. With this approach, we are going to transform our original process into two distinct processes with returns $g_N(x_N, y_N)$ and $\sum_{k=1}^{N-1} g_k(x_k, y_k)$ and maximize these two processes seperately.

The maximized return in open form is written as

$$\max_{\substack{(x_1,x_2,\ldots,x_N)\\(y_1,y_2,\ldots,y_N)}} \sum_{k=1}^{N} g_k(x_k,y_k). \tag{20}$$

Now, the first action we take is maximizing the results by first maximizing results on second resource, with an arbitrary allocation vector on the first resource, and after that, maximizing the return over the first resource. In short, the result we are looking for is

$$\max_{(x_1,x_2,\ldots,x_N)} \left\{ \max_{(y_1,y_2,\ldots,y_N)} \left\{ \sum_{k=1}^{N} g_k(x_k,y_k) \right\} \right\} \tag{21}$$

Next action to be taken is seperating the return of the initial stage, g_N, from the rest of the returns, which results in

$$\max_{(x_1,x_2,\ldots,x_N)} \left\{ \max_{(y_1,y_2,\ldots,y_N)} \left\{ g_N(x_N,y_N) + \sum_{k=1}^{N-1} g_k(x_k,y_k) \right\} \right\} \tag{22}$$

The value of g_N does not depend on the allocations of the second resource on the previous stages, but the value of $\sum_{k=1}^{N-1} g_k(x_k,y_k)$ depends on y_N; there will remain $\mathbf{y}-y_N$ much to allocate in the second dimension of the stages $1,2,\ldots,N-1$. Hence, optimization of $(y_1,y_2,\ldots,y_{N-1})$ will move inside the brackets, transforming the equation:

$$\max_{(x_1,x_2,\ldots,x_N)} \left\{ \max_{0\le y_N\le \mathbf{y}} \left\{ g_N(x_N,y_N) + \max_{\substack{(y_1,y_2,\ldots,y_{N-1}):\\ \sum_{k=1}^{N-1} y_k = y-y_N}} \left\{ \sum_{k=1}^{N-1} g_k(x_k,y_k) \right\} \right\} \right\}. \tag{23}$$

The same steps are repeated for the allocation of the first resource, and for problems with more types of resources, an iteration of this algorithm over each type of resource will suffice. Hence, the result for two resource allocation processes presents itself:

$$\max_{\substack{(x_1,\ldots,x_N)\\(y_1,\ldots,y_N)}} \sum_{k=1}^{N} g_k(x_k,y_k) = \max_{0\le x_N\le \mathbf{x}} \max_{0\le y_N\le \mathbf{y}} \left[g_N(x_N,y_N) + \max_{\substack{(x_1,\ldots,x_{N-1}):\\ \sum_{k=1}^{N-1} x_k = x-x_N}} \max_{\substack{(y_1,\ldots,y_{N-1}):\\ \sum_{k=1}^{N-1} y_k = y-y_N}} \sum_{k=1}^{N-1} g_k(x_k,y_k) \right]. \tag{24}$$

Set $f_n = f_n(\mathbf{x},\mathbf{y})$ to be the maximum return out of the first n stages of a two dimensional allocation process with resources $\mathbf{x}$ and $\mathbf{y}$. After seperating the return of the initial stage $g_N = g_N(x_N,y_N)$ from the rest of the return, we maximize both parts of the summation by choosing the right allocations $x_N \le \mathbf{x}$ and $y_N \le \mathbf{y}$. Since the second part of the summation is the maximum return from $n-1$ stages with the remaining resources $\mathbf{x}-x_n$ and $\mathbf{y}-y_n$, we are able to write f_n recursively:

$$f_n(\mathbf{x},\mathbf{y}) = \max_{0\le x_n\le \mathbf{x}} \max_{0\le y_n\le \mathbf{y}} \left[g_N(x_N,y_N) + f_{n-1}(\mathbf{x}-x_N,\mathbf{y}-y_N) \right]. \tag{25}$$

The principle of optimality trivially holds for one dimensional allocation processes as well.

Lagrange Multiplier

Even though the principle of optimality reduces the complexity of solving the problem, there's still lots of complexity arising from the fact that there are many dimensions to optimize the results over. Lagrange multiplier is a very intuitive technique with substantial effect on the complexity of multidimensional allocation processes.

Take an arbitrary allocation process with two types of resources of amounts $\mathbf{x}, \mathbf{y}$ and finitely many stages, and focus on the return of the first stage $g_1 = g_1(x_1, y_1)$. Fix an arbitrary allocation $(x_2, x_3, \dots)$ for the rest of the process, and set $(\mathbf{x}_1, \mathbf{y}_1)$ to be a local maximum of the return of the first stage, g_1. For simplicity, assume g_1 has a continuous second derivative over some open neighborhood of $(\mathbf{x}_1, \mathbf{y}_1)$. We would like to see the Taylor expansion for the function to observe the change in the value of g_1 when we move slightly away from $(\mathbf{x}_1, \mathbf{y}_1)$.

Take two values $h_1, h_2 > 0$ to control the changes in the parameters, and expand $g_1(\mathbf{x}_1 + h_1\Delta, \mathbf{y}_1 + h_2\Delta)$ around $(\mathbf{x}_1, \mathbf{y}_1)$ with respect to a small $\Delta > 0$(so that we stay in the open neighbourhood mentioned before):

$$\begin{aligned} g_1(\mathbf{x}_1 + h_1\Delta, \mathbf{y}_1 + h_2\Delta) &= g_1(\mathbf{x}_1, \mathbf{y}_1) + \frac{\partial g_1}{\partial x_1}(\mathbf{x}_1, \mathbf{y}_1)h_1\Delta + \frac{\partial g_1}{\partial y_1}(\mathbf{x}_1, \mathbf{y}_1)h_y\Delta + \mathcal{O}(\Delta^2) \\ &= g_1(\mathbf{x}_1, \mathbf{y}_1) + \Delta\left[h_1\frac{\partial g_1}{\partial x_1}(\mathbf{x}_1, \mathbf{y}_1) + h_2\frac{\partial g_1}{\partial y_1}(\mathbf{x}_1, \mathbf{y}_1)\right] + \mathcal{O}(\Delta^2). \end{aligned} \tag{26}$$

g_1 was assumed to have a local maximum on $(\mathbf{x}_1, \mathbf{y}_1)$, which tells us that partial slopes of g_1 are zero at $(\mathbf{x}_1, \mathbf{y}_1)$:

$$\frac{\partial g_1}{\partial x_1}(\mathbf{x}_1, \mathbf{y}_1) = \frac{\partial g_1}{\partial y_1}(\mathbf{x}_1, \mathbf{y}_1) = 0. \tag{27}$$

Recall that x_1 and y_1 are subject to the constrains 16 and 17 for the whole process:

$$x_1 = \mathbf{x} - \sum_{k\geq 2} x_k,\ y_1 = \mathbf{y} - \sum_{k\geq 2} y_k \tag{28}$$

This fact can be simply represented by the usage of a constant function $F = F(x_1, y_1) = (\mathbf{x} - \sum_{k\geq 2} x_k) + (\mathbf{y} - \sum_{k\geq 2} y_k)$, where values $\mathbf{x}$ and $\mathbf{y}$ are fixed. Then we deduce

$$\frac{\partial F}{\partial x_1}(\mathbf{x}_1, \mathbf{y}_1) = \frac{\partial F}{\partial y_1}(\mathbf{x}_1, \mathbf{y}_1) = 0. \tag{29}$$

Hence, we are able to write the following equations for some coefficient $\lambda \in \mathbb{R}$:

$$\begin{aligned} \frac{\partial g_1}{\partial x_1}(\mathbf{x}_1, \mathbf{y}_1) + \lambda\frac{\partial F}{\partial x_1}(\mathbf{x}_1, \mathbf{y}_1) = 0 \\ \frac{\partial g_1}{\partial y_1}(\mathbf{x}_1, \mathbf{y}_1) + \lambda\frac{\partial F}{\partial y_1}(\mathbf{x}_1, \mathbf{y}_1) = 0 \end{aligned} \tag{30}$$

The same equations appear when the function

$$G(\mathbf{x},\mathbf{y}) = g_1(\mathbf{x}_1,\mathbf{y}_1) + \lambda F = g_1(\mathbf{x}_1,\mathbf{y}_1) + \lambda(x_1 + y_1) \tag{31}$$

has a local maximum on the point $(\mathbf{x}_1, \mathbf{y}_1)$, without the constraints in 28.

The whole process is viable for the same actions since the outcome of the process is simply a linear combination of the stage returns, which are already represented as needed.Set $f_N(\mathbf{x},\mathbf{y})$ as in 25 with constraints 15, 16 and 17 on the resources $\mathbf{x},\mathbf{y}$. Instead of the constraint $0 \le y_N \le \mathbf{y}$, we are going to form a new function

$$\begin{aligned} P_N(\mathbf{x};\lambda) &= \max_{\substack{(x_1,x_2,\ldots,x_N)\\ \sum x_n = \mathbf{x}}} \left[\max_{(y_1,y_2,\ldots,y_N)} \left(g_1(x_1,y_1) + g_2(x_2,y_2) + \cdots + g_N(x_N,y_N) - \lambda \sum_{k=1}^{N} y_k \right) \right] \\ &= \max_{0\le x_N \le \mathbf{x}} \left[\max_{0\le y_N < \infty} \left(g_N(x_N,y_N) - \lambda y_N \right) + P_{N-1}(\mathbf{x} - x_N) \right] \end{aligned} \tag{32}$$

where we can intuit the Lagrange multiplier λ as the "cost" per one unit of the second resource. It is also possible to remove the constraint 16 by introducing another Lagrange multiplier θ and form the function

$$\begin{aligned} Q_N(\theta,\lambda) &= \max_{(x_1,x_2,\ldots,x_N)} \left[\max_{(y_1,y_2,\ldots,y_N)} \left(g_1(x_1,y_1) + \cdots + g_N(x_N,y_N) - \theta \sum_{k=1}^{N} x_k - \lambda \sum_{k=1}^{N} y_k \right) \right] \\ &= \max_{0\le x_N < \infty} \left[\max_{0\le y_N < \infty} \left(g_N(x_N,y_N) - \theta x_N - \lambda y_N \right) + Q_{N-1}(\mathbf{x} - x_N) \right]. \end{aligned} \tag{33}$$

Efficacy of Lagrange Multipliers Now we would like to see how the Lagrange multipliers allow us to obtain the optimal allocations for the initial problem. Before presenting the proof, we have to focus on a particularly important property of Lagrange multipliers. When we calculate the outcomes P_N in 32 with different values of λ, we obtain different amounts of total resource of type two allocated; this fact is very helpful when we intuit λ as "cost per unit resource". Namely, as λ increases, total allocated resource of type two decreases. In the calculations, this becomes very apparent as λ is iterated through many values. Keeping this in mind, we can obtain the specific value of $\lambda = \lambda(\mathbf{y})$ that results in the amount $\mathbf{y}$ allocated into the process through iteration. Further investigation will be a bit later on this fact. Now we can proceed to the proof.

Set $v_1 = (x_1^*, x_2^*, \ldots, x_N^*; y_1^*, y_2^*, \ldots, y_N^*)$ to satisfy $P_N(\mathbf{x};\lambda) = \sum_{k=1}^{N} g_k(x_k^*, y_k^*) - \lambda(\mathbf{y})\mathbf{y}$, i.e. assume v_1 is an allocation that results in the maximum return from the function Lagrange multiplier $\lambda = \lambda(\mathbf{y})$ is introduced in. For a contradiction, assume v_1 is not an optimal allocation for $\sum_{k=1}^{N} g_k(x_k, y_k)$, and set $v_2 = (x_1^+, x_2^+, \ldots, x_N^+; y_1^+, y_2^+, \ldots, y_N^+)$ to be the allocation that satisfies $f_N(\mathbf{x},\mathbf{y}) = g_1(x_1^+, y_1^+) + \cdots + g_N(x_N^+, y_N^+)$. Then we have

$$g_1(x_1^+, y_1^+) + \cdots + g_N(x_N^+, y_N^+) > g_1(x_1^*, y_1^*) + \cdots + g_N(x_N^*, y_N^*). \tag{34}$$

Since $\sum_{k=1}^{N} y_k^+ = \mathbf{y}$, total amount of resource of type two, used by both allocations, is equal:

$$\sum_{k=1}^{N} y_k^* = \sum_{k=1}^{N} y_k^+ = \mathbf{y} \tag{35}$$

Then the outcomes of these two allocations result in the inequality below when the lagrange multiplier is introduced for allocations of the second type:

$$\sum_{k=1}^{N} g_k(x_k^+, y_k^+) - \lambda \mathbf{y} > \sum_{k=1}^{N} g_k(x_k^*, y_k^*) - \lambda \mathbf{y} = P_N(\mathbf{x}, \mathbf{y}). \tag{36}$$

v_1 was assumed to be the solution of P_N but v_2 results in a larger outcome if v_1 is assumed to not maximize $g_1(x_1, y_1) + g_2(x_2, y_2) + \cdots + g_N(x_N, y_N)$, hence contradicting the basic assumption of v_1.

Choosing The Correct Lagrange Multiplier

The intuition of "cost" is very effective in visualizing the calculations.Trivially, infinite allocation under a cost results in infinite loss from the whole process. On top of this, we would like to see what happens to the amount of resources to be allocated when the price per unit resource increases.

Set two costs $\lambda_1 < \lambda_2$ on a process whose maximum return is given by 32 under constraints 15, 16 and 17. Assume $y_{\lambda_1} := (y_n^{(\lambda_1)})_{n=1}^N$ and $y_{\lambda_2} := (y_n^{(\lambda_2)})_{n=1}^N$ are two optimal allocations for the process whose outcomes are $G_{\lambda_1} := P_N(\mathbf{x}; \lambda_1)$ and $G_{\lambda_2} := P_N(\mathbf{x}; \lambda_2)$ with some arbitrary amount of first type of resource, $\mathbf{x}$. Then each of these vectors result in suboptimal outcomes under each others' λ values:

$$\sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_1)}) - \lambda_1 \sum_{k=1}^{N} y_k^{(\lambda_1)} \geq \sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_2)}) - \lambda_1 \sum_{k=1}^{N} y_k^{(\lambda_2)} \tag{37}$$

$$\sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_2)}) - \lambda_2 \sum_{k=1}^{N} y_k^{(\lambda_2)} \geq \sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_1)}) - \lambda_2 \sum_{k=1}^{N} y_k^{(\lambda_1)} \tag{38}$$

Subtract and add $\lambda_2 \sum y_k^{(\lambda_2)}$ to the right side of 37:

$$\begin{aligned} \sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_1)}) - \lambda_1 \sum_{k=1}^{N} y_k^{(\lambda_1)} &\geq \sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_2)}) - \lambda_2 \sum_{k=1}^{N} y_k^{(\lambda_2)} + \lambda_2 \sum_{k=1}^{N} y_k^{(\lambda_2)} - \lambda_1 \sum_{k=1}^{N} y_k^{(\lambda_2)} \\ &\geq \sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_2)}) - \lambda_2 \sum_{k=1}^{N} y_k^{(\lambda_2)} + (\lambda_2 - \lambda_1) \sum_{k=1}^{N} y_k^{(\lambda_2)} \end{aligned} \tag{39}$$

The left side of equation 38 has appeared in 39. Hence,

$$\sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_1)}) - \lambda_1 \sum_{k=1}^{N} y_k^{(\lambda_1)} \geq \sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_1)}) - \lambda_2 \sum_{k=1}^{N} y_k^{(\lambda_1)} + (\lambda_2 - \lambda_1) \sum_{k=1}^{N} y_k^{(\lambda_2)}. \tag{40}$$

Removing $\sum_{k=1}^{N} g_k(x_k, y_k^{(\lambda_1)})$ from both sides and sending $\lambda_2 \sum_{k=1}^{N} y_k^{(\lambda_1)}$ to the left side, we obtain

$$(\lambda_2 - \lambda_1) \sum_{k=1}^{N} y_K^{(\lambda_1)} \geq (\lambda_2 - \lambda_1) \sum_{k=1}^{N} y_k^{(\lambda_2)} \tag{41}$$

.

We set λ_1 to be smaller than λ_2, so dividing both sides with $\lambda_2 - \lambda_1$ results in

$$\sum_{k=1}^{N} y_k^{(\lambda_1)} \geq \sum_{k=1}^{N} y_k^{(\lambda_2)} \tag{42}$$

As expected, an increase in the cost of one resource cannot lead to a larger allocation of that resource. The amount of resource $\mathbf{y}$ that's expended on the processes will be given when we are solving problems, so iterating different values of λ will result in various different optimal allocations of the related resource. For a given amount of resource $\mathbf{y}$, the Lagrange multiplier $\lambda = \lambda(\mathbf{y})$ that results in an allocation $(y_k)_{k=1}^{N}$ with $\sum_{k=1}^{N} y_k = \mathbf{y}$ will be fixed to figure out the solutions for our original processes.

The Flyaway-kit Problem [1]

We are going to investigate another cargo problem. Say we are in charge of a transport plane that's going to carry replacement parts for planes in an overseas base. For every travel, the base reports a certain number of demand for N different types of items, and if the demand is not supplied, there incurs a cost per missing item. For example, if 3 items of type 1 is needed at the base, and there's only one item of type 1 carried to the base with our transport plane, we will be obliged to pay 2 times the given penalty for type 1 items.

The transport plane has a certain capacity for the total weight and volume of the items we allocate, so the total amount of items we can carry is limited. Our aim is to find the optimal choice of items to be carried under the weight and size constraints of our carrier.

The demand for each type of items is assumed to have a Poisson distribution over non-negative integers with expected values values λ_n for each type $n = 1, 2, \ldots, N$. As well as this, there will be non-negative costs c_n, weights w_n and sizes s_n for each type of item n. We will construct allocation vectors $(a_1, a_2, \ldots, a_n)$ where each entry a_n shows how many items of type n has been placed into our transport plane, and each allocation vector will obey the constraints

1. There is no negative allocation, i.e. for every item n, $a_n \geq 0$.
2. Sum of the weights have to be less than or equal to the total weight capacity $\mathbf{w}$ of the transport plane, i.e. $\sum_{k=1}^{N} a_k w_k \leq \mathbf{w}$.
3. Sum of the sizes have to be less than or equal to the total size capacity $\mathbf{s}$ of the transport plane, i.e. $\sum_{k=1}^{N} a_k s_k \leq \mathbf{s}$.

Create an allocation $(a_1, a_2, \ldots, a_N)$ with the constraints mentioned above. Then for each transport the plane will carry a_n items of type $n = 1, 2, \ldots, N$. The occurence of a cost implies that for some types of items the demand was larger than the allocated number. Set $X_n \sim Pois(\lambda_n)$ to be the demand for items of type n at the base. If x_n is less than or equal to a_n, then the demand is supplied, and no cost occurs. Otherwise, a cost of $c_n(x_n - a_n)$ will occur. On average, we expect a cost

$$\sum_{i \geq a_n + 1} c_n(i - a_n)\mathbb{P}(i; \lambda_n) \tag{43}$$

from type n per trip to the base when a_n many items of type n is allocated, where $\mathbb{P}(i; \lambda_n)$ is the probability of the demand being equal to i under the average demand λ_n.

Our aim is to find the optimal allocation of various types of items into our transport plane so that we have the least total expected cost for each transport. By using the dynamic programming principles, we allocate each type of item one by one at each stage, and maximizing the returns by iterating over different amounts of allocations. At the N'th stage of this process, the optimal cost is given by the equation

$$f_N(\mathbf{x}, \mathbf{y}) = \min_{a_N \in \{0,1,\ldots,m\}} \left[\sum_{k \geq a_N + 1} c_N(k - a_N)\mathbb{P}(k; \lambda_N) + f_{N-1}(\mathbf{x} - w_N a_N, \mathbf{y} - s_N a_N) \right] \tag{44}$$

where m is the maximum number of items of type N possible to put in the transport plane.

Even though there's only one dimension in the allocations, which is the number of items of type N to be allocated, the calculations require us to iterate over different capacities of weights for $\mathbf{x}$ and size $\mathbf{y}$. Consider the case where the size is not considered as a part of the problem, that is in turn a one dimensional allocation process. Assume that the time it takes to run the main loop of this case, which is iterated by different values of capacities, is the same with that of the original problem. If the case where size is not considered is iterated for more than 10^{2k} different values of weight from 0 to $\mathbf{x}$, then our original problem can be iterated over around 10^k different values on both resource values for the calculations of these two processes to take the same amount of time. It's easy to see how this translates to problems with more dimensions in the allocations. For this reason, in the calculations, loops were iterated for weights and sizes up to 30.

Results Below are the plots that show the optimized costs of the procedures with certain alignments of weight and size capacities:

The outcomes on [1, Page 45] are slightly off from my calculations. To make sure whose calculations were true, I calculated the expected cost for weight and size capacity equal to 2. The optimal allocation of items in this case is obtained by putting only one item of type 6, which has a weight and size equal to 2. Other 9 type of items are not allocated, so the expected

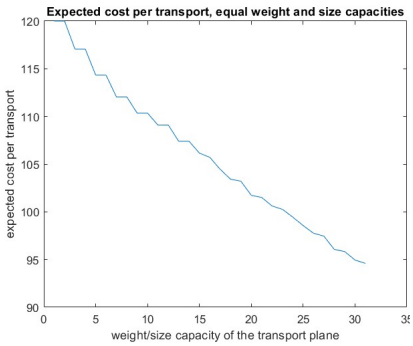

Figure 3: *The cost from transporting the optimal selection of replacement parts, where equal weight and size is assumed on the transport plane*

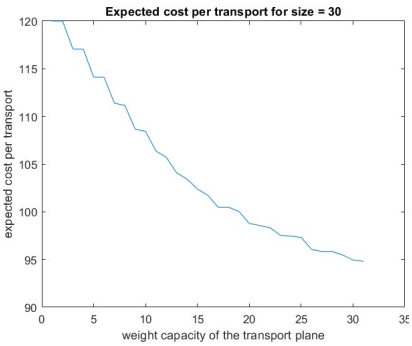

Figure 4: *The cost from transporting the optimal selection of replacement parts for different weight capacities at size=30 on the transport plane*

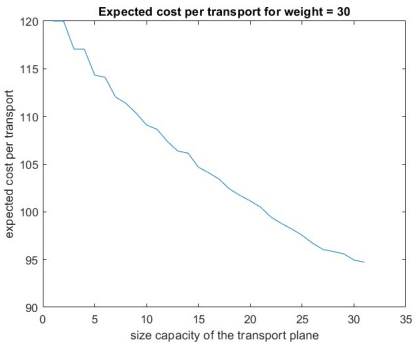

Figure 5: *The cost from transporting the optimal selection of replacement parts for different size capacities at weight=30 on the transport plane*

cost from those 9 items is 9 times the expected demand times the cost, which is equal to 108. With the cost due to having one replacement part of type 6, we obtain a total cost

$$\begin{aligned}
&108 + \sum_{i\geq 2} c_6(i-1)\mathbb{P}(i;\lambda_6) = 108 + 3\sum_{i\geq 2}(i-1)\mathbb{P}(i;4) = 108 + 3\left[1\mathbb{P}(2;4) + 2\mathbb{P}(3;4) + 3\mathbb{P}(4;4) + \dots\right] \\
&= 108 + 3\cdot(0.1465 + 2\cdot 0.1954 + 3\cdot 0.1954 + 4\cdot 0.1563 + 5\cdot 0.1042 + 6\cdot 0.0595 + 7\cdot 0.0298 \\
&\quad + 8\cdot 0.0132 + 9\cdot 0.0053 + 10\cdot 0.0019 + 11\cdot 0.0006 + 12\cdot 0.0002 + 13\cdot 0.00005 + \dots) \\
&\approx 108 + 3\cdot 3.01725 = 117.05175,
\end{aligned} \tag{45}$$

by the equation 43. In [1, Page 45] this has been calculated as 116.1. Similarly, the expected cost when the weight and size capacities are equal to 30 is calculated to be 88.8, whereas my calculations for the same allocation of resources is around 94.

Advertising Campaign Problem [1]

Assume that you are responsible for the operations of N departments in a company and every department admits a certain return on the activities depending on the allocation of production and advertisement investments. The return from the i'th department when a production budget x_i and an advertisement budget y_i (from [1]) is assumed to be

$$g_i(x_i, y_i) = \alpha_i\left[1 - (1 - e^{-\frac{\beta_i}{x_i+y_i}})^{x_i}\right] \tag{46}$$

where α_i is the maximum theoretical return from investing into the i'th department, and β_i is the rate of competition in the market of the i'th department. Even though x_i and y_i cannot be equal to zero at the same time, this is not the biggest problem in the calculations since division by 0 is encoded to be `Inf`. Hence, we are looking for the allocation of total production budget

$\mathbf{x}$ and total advertisement budget $\mathbf{y}$ that maximizes the return $\sum_{k=1}^{N} \alpha_k \left[1 - (1 - e^{-\frac{\beta_k}{x_k + y_k}})^{x_k}\right]$ under the constraints

$$\sum_{k=1}^{N} x_k \leq \mathbf{x}, \tag{47}$$

$$\sum_{k=1}^{N} y_k \leq \mathbf{y}, \tag{48}$$

and for all $k = 1, 2, \ldots, N$

$$x_k, y_k \geq 0. \tag{49}$$

Define $f_N(\mathbf{x}, \mathbf{y})$ to be the function that outputs the maximum obtainable outcome out of distributing the production budget $\mathbf{x}$ and advertisement budget $\mathbf{y}$ onto N many departments. By the principle of optimality, we write the recursive relation

$$f_N(\mathbf{x}, \mathbf{y}) = \max_{0 \leq x_N \leq \mathbf{x}} \max_{0 \leq y_N \leq \mathbf{y}} \left[g_N(x_N, y_N) + f_{N-1}(\mathbf{x} - x_N, \mathbf{y} - y_N)\right] \tag{50}$$

Now we are ready to introduce the Lagrange multiplier into the equation. Recall that we can swap the constraints on one of the resources 47 or 48 by attaining a "cost" on the resource. In this spirit, we remove the upper limit $\mathbf{y}$ for the advertisement budget and add a cost per investment in advertisement, to later find the "correct cost", which incurs the total advertisement budget to be equal to $\mathbf{y}$. We already showed that as the cost per investment increases, the amount of investment either stays the same or gets lowered by an increase in the cost.

Results Figure 6 to figure 9 show how the advertisement budget is affected by the production budget.

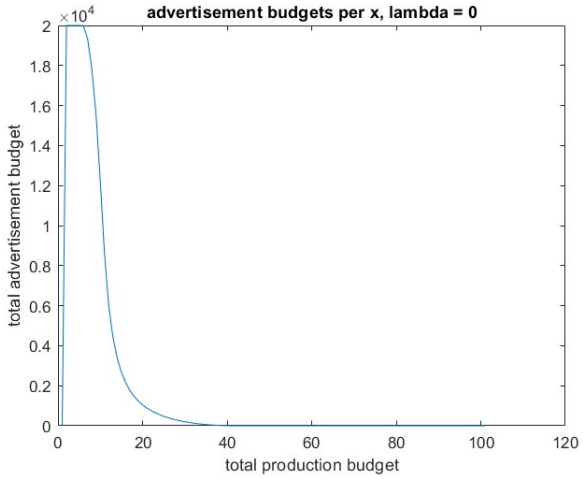

Figure 6: *Total investment to advertisement for different production budgets, with zero cost per unit advertisement budget*

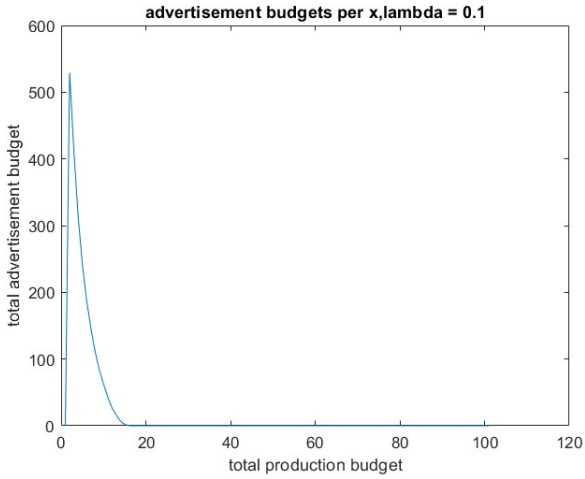

Figure 7: *Total investment to advertisement for different production budgets, with a cost of* 0.1 *per unit advertisement budget*

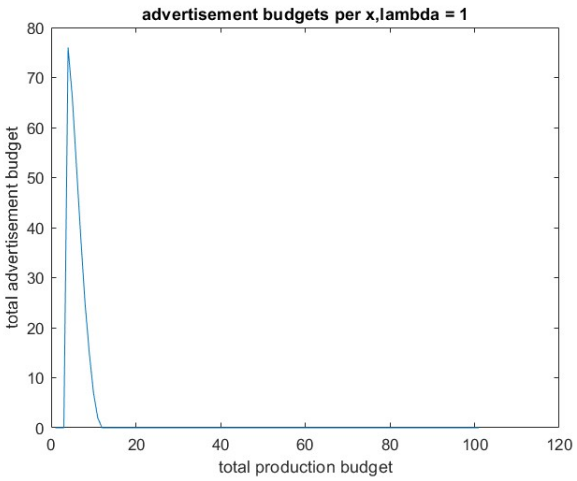

Figure 8: *Total investment to advertisement for different production budgets, with a cost of* 1 *per unit advertisement budget*

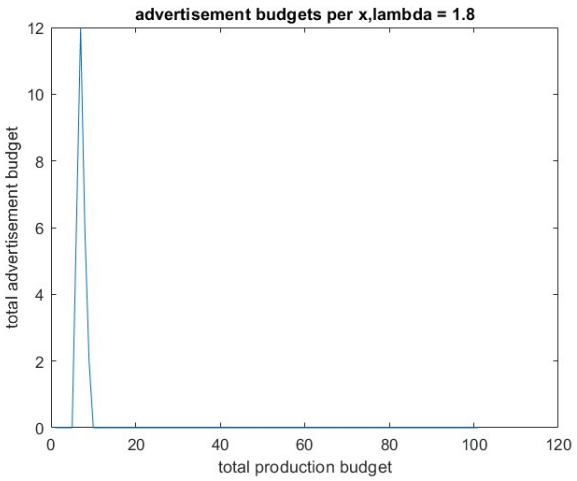

Figure 9: *Total investment to advertisement for different production budgets, with a cost of* 1.8 *per unit advertisement budget*

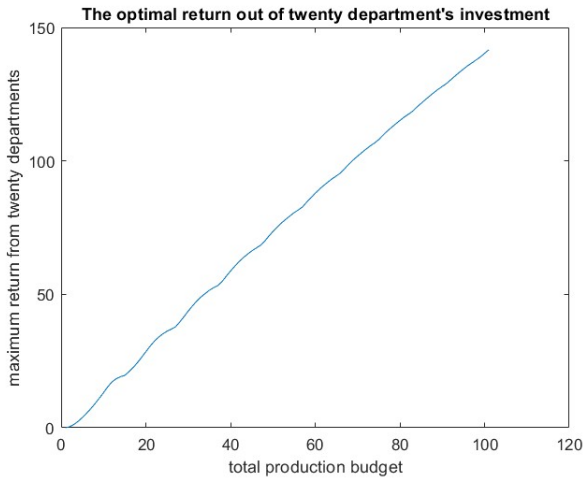

Figure 10: *Returns from the optimal investments of production and advertisement budgets into twenty different departments*

Figure 10 shows the plot for the optimal outcomes from allocating different total production budgets to twenty different departments where department $i = 1, 2, \ldots, 20$ has maximum obtainable profit and rate of competition equal to i and cost per unit advertisement investment is 1.8

Recall that as costs per unit advertisement investment increases, the total budget for advertisement reduces. Figure 10 is constructed by using the outcomes of calculations of the equation 32. For $\lambda = 1.8$, we see that total advertisement budget is zero from figure 9 whenever the production budget is greater than 20. Hence, the plot figure 10 shows the outcomes for our original process as well.

Figure 11 shows the best production investments towards each department, from 1 to 20, for total production budgets 50, 75 and 100.

department:	1 to 9	10	11	12	13	14	15	16	17	18	19	20	returns
pb:50	0	0	0	0	0	0	0	9	10	10	10	11	75.1505
pb:75	0	0	0	0	8	8	9	9	10	10	10	11	109.5811
pb:100	0	7	7	8	8	9	9	10	10	10	11	11	141.6464

Figure 11: *The optimal allocations of different production budgets onto twenty departments, with zero advertisement investments*

Markovian Decision Processes [1]

Now we focus on processes where at each stage we obtain an element in the state space and choose strategies/control elements to dictate a Markov chain, so to maximize the expected outcome of each transition. For each process of this nature, we will assume a finite state space, whose elements will simply be referred by indices $i = 1, 2, \ldots, N$. Consider such a process with finitely many stages and initial state i from a finite state space $A = \{1, 2, ...N\}$. We are going to form a control vector, where each entry of this vector comes from a finite set of choices $C_n(i) \subset C = \{\mathbf{1}, \mathbf{2}, ..., \mathbf{k}\}$ applicable for state $i = 1, 2, \ldots, N$ at the n'th stage of the process. After constructing a vector

$$u_n = (c_1^{(n)}, c_2^{(n)}, ..., c_N^{(n)}) \tag{51}$$

where each $c_i^{(n)}, i = 1, 2, ..., N$ comes from $C_n(i)$ and represents the control element chosen at stage n when the initial state is i, we obtain a set of pairs that come with u_n:

$$S_n(u_n) = \left\{(\pi_n(c_i^{(n)}), \rho_n(c_i^{(n)})) : i = 1, 2, ..., N\right\}. \tag{52}$$

Control outcomes $(\pi_n(c_i^{(n)}), \rho_n(c_i^{(n)}))$ are pairs of probability mass functions $\pi_n(c_i^{(n)})$ that measure the likelihood of transitioning to one of the states $j = 1, 2, ..., N$ at the n'th stage of the process given that the initial state is i, and an associated stage return vector $\rho_n(c_i^{(n)})$ that shows the amount to be obtained when transitioned to states $j = 1, 2, ..., N$:

$$\pi_n(c_i^{(n)}) = \left(p_{ij}(c_i^{(n)})\right)_{j=1}^{N}, \rho_n(c_i^{(n)}) = \left(r_{ij}(c_i^{(n)})\right)_{j=1}^{N} \tag{53}$$

where $p_{ij}(c_i^{(n)})$ is the likelihood of transitioning to state j starting from state i under the control element $c_i^{(n)}$ from the set $C_n(i)$ of attainable control elements at stage n at state i, and $r_{ij}(c_i^{(n)})$ is the amount to be obtained when transitioned to state j from state i. Recall that these values are dictated by the control element $c_i^{(n)}$ we chose to apply when we are at state i at the n'th stage of our process.

When the transition vectors $\pi_n(c_i^{(n)})$ and $\rho_n(c_i^{(n)})$ are row-binded elementwise, we obtain a Markovian transition probability matrix $P(u_n)$, as well as an associated return matrix $R(u_n)$. Together, they dictate the transition to the $n+1$'th stage of our process.

Recursive Definition of Returns

With the Markovian matrices formed above, we can formulate the expected outcome of a process with a recursion. Set $f_n(i)$ to be the "optimal" expected outcome of an n-stage Markovian decision process at initial state i with the control vectors $u_k : k = 1, 2, ..., n$ to be applied at every stage.

After choosing a control vector $u_n := (c_1, c_2, \dots, c_N)$ to be applied at n'th stage while starting at state i, we have a certain likelihood of reaching state j that is dictated by the pmf $\pi_n(c_i) = (p_{ij}(c_i))_{j=1}^N$. In other words, when starting at state i at the initial stage, we have $p_{ij}(c_i)$ likelihood of transitioning to state j. Similarly, there will be a return $r_{ij}(c_i)$ to be obtained from this transition. After this, there will be $n-1$ stages remaining in our hand, and our initial state will be j. In this case, we obtained $r_{ij}(c_i) + f_{n-1}(j)$ from the whole process, given that we transitioned to state j.

Hence, the optimal result $f_n(i)$ to be obtained from our n-stage process, starting from state i, is formulated as

$$\begin{aligned} f_n(i) &= \max_{c_i \in C_n(i)} \sum_{j=1}^{N} p_{ij}(c_i) \left[r_{ij}(c_i) + f_{n-1}(j) \right] \\ &= \sum_{j=1}^{N} p_{ij}(c_i^*) \left[r_{ij}(c_i^*) + f_{n-1}(j) \right]. \end{aligned} \tag{54}$$

where c_i^* is the optimal control element which $f_n(i)$ is defined on. It is also convenient to form only one control vector/policy, and so to be alleviated from writing superscripts on the control elements as in the previous subsection.

Since n is chosen arbitrarily, processes with large number of steps and approximations of infinite processes are available for analysis under following theorem [1, page 301]:

Theorem 1 *Define a recursive function*

$$f_n(j) = \max_{c_j \in C_n(j)} \left[b_j(c_j) + \sum_{i=1}^{N} a_{ij}(c_i) f_{n-1}(i) \right] \tag{55}$$

dependent on the control element c_j with the assumptions

1. *For any $j = 1, 2, ..., N$, $b_j(c_j) \geq 0$.*
2. *For at least one of the states $j \in \{1, 2, ..., N\}$, $b_j(c_j) > 0$.*
3. *For all $i, j = 1, 2, ..., N$, $a_{ij}(c_j)$ is strictly positive.*
4. *$\sum_{j=1}^{N} a_{ij}(c_j) = 1$.*

Further assume for b_j and a_{ij} that either both are continuous, or both have finite range. And in the case of continuity, assume the image sets of b_j and a_{ij} are closed and bounded.

Let $\mathbf{I} = (1, 1, ..., 1)$ *of length* N, $\mathbf{b}(u_n) = (b_1(c_1), b_2(c_2), \dots, b_N(c_N))$ *and* $\mathbf{A}(u_n) = (a_{ij}(c_i))_{i=1,j=1}^{N}$. *Then as* $n \to \infty$,

$$f_n(i) \sim nr, i = 1, 2, ..., N \tag{56}$$

where the scalar r *is determined by*

$$r\mathbf{I} = \max_{u_n} \lim_{n\to\infty} \left[\frac{\mathbf{b}(u_n) + \mathbf{A}(u_n)\mathbf{b}(u_n) + \cdots + \mathbf{A}(u_n)^{n-1}\mathbf{b}(u_n)}{n} \right] \tag{57}$$

A proof of this theorem is found on [3].

Policy Space Technique

In theorem 1, the function defined by the recurrence relation

$$f_n(j) = \max_{c_j \in C_n(j)} \left[b_j(c_j) + \sum_{i=1}^{N} a_{ij}(c_i) f_{n-1}(i) \right] \tag{58}$$

is shown to be approximately nr for some real number r and for large values of n when the recurrence relation is followed. With this in mind, we will introduce a new technique called "policy space technique" [1, Page 303].

Assume we are working on an n-stage Markovian decision process for some large natural number n who satisfies the constraints in the theorem 1. We can think of such processes as strictly state-bound, as in increasing the number of stages is going to increase the total return almost exactly by a certain expected amount, independent of the specific stage the process is at. For this process, we will define V_i^n to be the total expected return from n stages,starting from an arbitrary state $i = 1, 2, \dots, N$ under a fixed policy/control vector $u = (c_1, c_2, \dots, c_N)$, where N denotes the number of states in this process. Then

1. We can write the recursive relation between consecutive stages of the process with total expected returns V_i^n by applying the dynamic programming technique. Since we have a control element c_i chosen to be applied at state i, we arrive at state j and obtain a return from transitioning to state j via the probability distribution $\pi(c_i)$ and return vector $\rho(c_i)$ dictated by c_i. Defining $b_i(c_i) = \sum_{j=1}^{N} p_{ij}(c_i) r_{ij}(c_i)$ to be the average gain of an arbitrary stage n due to starting from state i with respect to the control element c_i, the recursive relation will be built on seperating the total expected return of n stages into the average gain from the initial stage n and the total expected return from the remaining $n-1$ stages

$1, 2, \ldots, n-1$. Hence,

$$\begin{aligned} V_i^n = \sum_{j=1}^{N} p_{ij}(c_i) \left[r_{ij}(c_i) + V_j^{n-1} \right] &= \sum_{j=1}^{N} p_{ij}(c_i) r_{ij}(c_i) + \sum_{j=1}^{N} p_{ij}(c_i) V_j^{n-1} \\ &= b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i) V_j^{n-1}. \end{aligned} \tag{59}$$

2. Observe that there exists a scalar g that is going to appear as the average expected return of one stage in our process due to the theorem 1. Since the process will continue for n stages, on average we expect to gain ng total return. What's also important is that since it's indeterministic what's going to happen after the initial stage, we are only able to infer the return from the initial state i. In light of these information, we can write V_i^n as

$$V_i^n = v_i + (n-1)g \tag{60}$$

where the term v_i is the return from the initial state when the initial state is i, and $(n-1)g$ is the total expected return of the remaining $n-1$ stages of our process.

In light of these two expressions, we can obtain a representation of g in terms of v_i's and control elements c_j:

1.
$$V_i^n = \sum_{j=1}^{N} p_{ij}(c_i) \left[r_{ij}(c_i) + V_j^{n-1} \right] \tag{61}$$

2.
$$V_i^n = v_i + (n-1)g \; \forall i = 1, 2, \ldots, N \tag{62}$$

3. For any transitioned state j at the $n-1$'th stage, we will have a return (from the remaining stages of the process)
$$V_j^{n-1} = v_j + (n-2)g \tag{63}$$

4.
$$\begin{aligned} v_i + (n-1)g &= \sum_{j=1}^{N} p_{ij}(c_i) r_{ij}(c_i) + \sum_{j=1}^{N} p_{ij}(c_i) \left(v_j + (n-2)g \right) \\ &= b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i) \left(v_j + (n-2)g \right) \\ &= b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i) v_j + (n-2)g \sum_{j=1}^{N} p_{ij}(c_i) \\ &= b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i) v_j + (n-2)g \end{aligned} \tag{64}$$

5. Cancelling out $(n-2)g$ from both sides, we obtain

$$v_i + g = b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i) v_j \tag{65}$$

6. And finally, we subtract the return of the initial stage, starting at state i to obtain the average return of n stages in terms of state nominal effects and control elements:

$$g = b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i)v_j - v_i \tag{66}$$

Now, values $p_{ij}(c_i)$ and $r_{ij}(c_i)$ are given by the fixed policy. For each $i = 1, 2, \ldots, N$, we have unknowns v_i, as well as g. In each of these N equations (of the form 5 above) contain g, so g itself being unknown creates a bottleneck of having $N + 1$ variables with N equations. A solution for this is proposed by Bellman relying on the fact below:

Set d to be an arbitrary real number. The system of N equations presented above are not disturbed when v_i's are replaced with $v_i + d$:

$$\begin{aligned} b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i)(v_j + d) - (v_i + d) &= b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i)v_j + \sum_{j=1}^{N} p_{ij}(c_i)d - (v_i + d) \\ &= b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i)v_j - v_i + d - d \\ &= b_i(c_i) + \sum_{j=1}^{N} p_{ij}(c_i)v_j - v_i = g \end{aligned} \tag{67}$$

What this fact shows us is, the "real" values of v_i's are not important since the average return per stage is not affected by changes in v_i's' value. Now, Bellman and Dreyfus' solution is setting one of v_i's to be 0, to represent the "default state", or a "benchmark" for returns from each state. For example. say we map each v_i to $v_i - v_N$. v_N is now equal to 0, so the number of unknown variables has dropped to N, solving the system of N equations.

Hence, we can see the average return $g = g(u)$ of each stage under the fixed policy $u = (c_1, c_2, \ldots, c_N)$, as well as the relative returns $v_i = v_i(u), i = 1, 2, ..., N$ of starting from different states, when we set $v_N = 0$.

The Policy Improvement Routine [1, Page 304]

All the values obtained in the previous section are obtained via only one control vector/policy u, which was arbitrarily chosen. Our main aim is to find the fixed policy $u^* = (c_1^*, c_2^*, c_3^*)$ that maximizes the average return per stage, $g = g(u^*)$, so that for each state $i = 1, 2, \ldots, N$, the total expected return of the n-stage process $V_i^n = v_i + (n-1)g$ is maximized. Recall that we accept this last equality when n is sufficiently large. For large n, the return from the initial stage is negligible, i.e.

$$\lim_{n\to\infty} \frac{v_i}{v_i + (n-1)g} = 0. \tag{68}$$

By using this fact, the policy improvement routine is introduced [1, Page 304] as an iterative way of finding a better fixed policy $u^p, p \in \mathbb{N}$ while using previous policy outcomes $v_i(u^{p-1})$.

Assume that we have been applying a policy u on a process with large number of stages n, and we have obtained an average gain $g(u)$ per stage together with $v_1(u), v_2(u), \ldots, v_N(u)$. At each state i, we can look at the set of applicable controls $C_n(i) = C(i) \subset C = \{\mathbf{1}, \mathbf{2}, \ldots, \mathbf{k}\}$ to later obtain a new policy $u^1 = (c_1^1, c_2^1, \ldots, c_N^1)$ that maximizes the right-hand side of (66) with the new probability distribution $\pi^1(u^1)$ and return vector $\rho^1(u^1)$.

Say $g(u^1) \geq g(u)$.

1. If $g(u^1) = g(u)$, then it means we obtained a policy that has resulted in no increase in average gain, giving us no reason to adjust to it. Hence, we set $u^1 = u$ to later obtain the same total expected return $v_i + (n-1)g$ from n stages.
2. If $g(u^1) > g(u)$, then

$$(n-1)\Big[g(u^1) - g(u)\Big] \to \infty \text{ as } n \to \infty, \tag{69}$$

which implies that there is a natural number $K \in \mathbb{N}$ such that

$$K\Big[g(u^1) - g(u)\Big] \geq v_i(u) - v_i(u^1), \tag{70}$$

and so

$$v_i(u^1) + Kg(u^1) \geq v_i(u) + Kg(u). \tag{71}$$

Hence, whenever we have $n > K$ stages,

$$v_i(u^1) + (n-1)g(u^1) \geq v_i(u) + (n-1)g(u) \tag{72}$$

for any state i.

The policy improvement routine can be intuited as simply partaking in n stages, observing the results, and on top of the information that's collected, such as the initial stage returns for each state, and looking for other policies that have the potential of allowing for higher average return with the prior information.

Taxicab Example [1]

Our story in this section starts with a taxi driver who drives between three towns. At each town, our driver is trying to choose the best action to find a customer who wants to travel to one of the three towns. Different customers have different lifes and urgencies, so the location a passenger is taken by our taxi driver has implications on where the next destination our dear taxi driver will arrive to. The taxi driver is aiming to obtain maximum return from each ride, and to make this possible, they have various different strategies to apply:

- Cruising (**1**): Strolling around in the streets to hopefully find a customer in rush,
- Going to a cab stand (**2**): Waiting in line with other taxi drivers for customers,

- Pulling to a curb (**3**): Waiting for a radio call for the next customer.

Each of these actions, when chosen on a town i, $i = 1, 2, 3$, present a probability mass distribution and a return vector for transitioning to town j, $j = 1, 2, 3$. State 2 does not allow for radio calls, so if the taxi driver lands in state 2 after a travel, then they have two options of action: cruising or going to a cab stand.

Below are given the outcome of the strategies [1, Page 300]:

```
state_1_prob = [ [1/2 1/4 1/4]; [1/16 3/4 3/16]; [1/4 1/8 5/8] ];
state_2_prob = [ [1/2 0 1/2]; [1/16 7/8 1/16] ];
state_3_prob = [ [1/4 1/4 1/2]; [1/8 3/4 1/8]; [3/4 1/16 3/16] ];
```

```
state_1_return = [ [10 4 8]; [8 2 4]; [4 6 4] ];
state_2_return = [ [14 0 18]; [8 16 8] ];
state_3_return = [ [10 2 8]; [6 4 2]; [4 0 8] ];
```

For an explanation of the construction of the transition matrices, assume we choose to cruise on each state. Then we get the first elements in the vectors above with respect to the policy/control vector $u = (\mathbf{1}, \mathbf{1}, \mathbf{1})$. If instead we had $u = (\mathbf{2}, \mathbf{1}, \mathbf{3})$, we would pick the second elements from `state_1_prob` and `state_1_return`, first elements from `state_2_prob` and `state_2_return`, and so on.

Figure 12 and figure 13 are matrices $P = P(u)$ that shows the probabilities of transitioning between stages and $R = R(u)$ that shows the returns of those transitions, respectively.

$$P = \begin{bmatrix} 0.5000 & 0.2500 & 0.2500 \\ 0.5000 & 0 & 0.5000 \\ 0.2500 & 0.2500 & 0.5000 \end{bmatrix} \quad (73)$$

Figure 12: *The transition probabilities between the three towns when the taxi drives chooses to cruise in the streets with the hope of finding a new customer.*

$$R = \begin{bmatrix} 10 & 4 & 8 \\ 14 & 0 & 18 \\ 10 & 2 & 8 \end{bmatrix} \quad (74)$$

Figure 13: *The expected returns of taking a customer the three towns when the taxi drives chooses to cruise in the streets with the hope of finding a new customer.*

It should be noted that u is a fixed policy; the transitions will be dictated by P and R at each stage of our process. The future discourse will explain how to navigate in the policy space once an initial element is obtained.

Recall that we read these matrices as such: the rows represent the state we currently are in, and columns represent the next possible state. The entries of P, p_{ij}, and R, r_{ij}, represent the transition from state i to the state j.

Solving The Linear System For Average Gain and Initial State Returns Now we will disect the relations between the average gain per stage g and expected returns of each stage v_i by using the Policy Space Technique. First, we obtain $b_i(\mathbf{1})$, the average outcome of transitioning from state i, by averaging returns of transition by their likelihood of occuring, which means multiplying P and transpose of R and obtaining the diagonal entries:

$$PR^T = \begin{bmatrix} 0.5000 & 0.2500 & 0.2500 \\ 0.5000 & 0 & 0.5000 \\ 0.2500 & 0.2500 & 0.5000 \end{bmatrix} \times \begin{bmatrix} 10 & 14 & 10 \\ 4 & 0 & 2 \\ 8 & 18 & 8 \end{bmatrix} = \begin{bmatrix} \mathbf{8} & \frac{23}{2} & \frac{15}{2} \\ 9 & \mathbf{16} & 9 \\ \frac{15}{2} & \frac{25}{2} & \mathbf{7} \end{bmatrix} \tag{75}$$

Recall that P and R contain the transition vectors in the rows; instead of calculating expected returns for each state, we can get those outcomes with PR^T. Hence, we are going to obtain 8 units on average if we take a customer from town 1,16 units if we take a customer from town 2, and 7 units if we take a customer from town 3.

Let's assume we take a customer in town 1 to bring them to their target. Since we don't know which town we will land in, we will look at the expected return to come from the next stage. We will stay in town 1 with likelihood $\frac{1}{2}$, and if so, the next stage is expected to return v_1 units. Similarly, we will go to town 2 with likelihood $\frac{1}{4}$, and if so, the next stage will on average return v_2 units. Hence, the return $g + v_i$ from two consecutive stages is obtained by the equations below:

$$v_1 + g = b_1(\mathbf{1}) + \sum_{j=1}^{3} p_{1j}(\mathbf{1})v_j = \mathbf{8} + \frac{1}{2}v_1 + \frac{1}{4}v_2 + \frac{1}{4}v_3 \tag{76}$$

$$v_2 + g = b_2(\mathbf{1}) + \sum_{j=1}^{3} p_{2j}(\mathbf{1})v_j = \mathbf{16} + \frac{1}{2}v_1 + 0v_2 + \frac{1}{2}v_3 \tag{77}$$

$$v_3 + g = b_3(\mathbf{1}) + \sum_{j=1}^{3} p_{3j}(\mathbf{1})v_j = \mathbf{7} + \frac{1}{4}v_1 + \frac{1}{4}v_2 + \frac{1}{2}v_3 \tag{78}$$

After obtaining the system of 3 linear equations with 4 independent variables, we take advantage of the fact that relative values of initial state returns can solve the system by reducing the number of equations by 1. Hence, we set $v_3 = 0$ and rewrite the 3 equations of 3 variables:

$$v_1 + g = b_1(\mathbf{1}) + \sum_{j=1}^{3} p_{1j}(\mathbf{1})v_j = 8 + \frac{1}{2}v_1 + \frac{1}{4}v_2 \tag{79}$$

$$v_2 + g = b_2(\mathbf{1}) + \sum_{j=1}^{3} p_{2j}(\mathbf{1})v_j = 16 + \frac{1}{2}v_1 + 0v_2 \tag{80}$$

$$g = b_3(\mathbf{1}) + \sum_{j=1}^{3} p_{3j}(\mathbf{1})v_j = 7 + \frac{1}{4}v_1 + \frac{1}{4}v_2 \tag{81}$$

In order to solve this system easily, we will make use of the function `linsolve` embedded in MATLAB by representing the outcomes of setting $v_3 = 0$. We have to take care of some remarks, though:

1. First, realize that setting $v_3 = 0$ makes the final column of P useless in further calculations; this matrix is the probability matrix for transitioning between towns, so the final column is the likelihood of transitioning to town 3, and each value in this column is multiplied by 0 in the equations.
2. Each equation contains the variable g exactly once.

In light of these two facts, we will first form the matrix that represents the system of equations, to later solve for g, v_1, v_2. For this, we represent the system as

$$Ax = b, \tag{82}$$

where A is a matrix that contains the coefficients from the system of equations that work on $x = (g, v_1, v_2)$, and $b = (8, 16, 7)$ is the outcome of the system. To find A, we subtract terms with v_1, v_2 in from both sides, so that on the right side of the system, we only will have b:

$$g + \frac{1}{2}v_1 - \frac{1}{4}v_2 = 8 \tag{83}$$

$$g - \frac{1}{2}v_1 + v_2 = 16 \tag{84}$$

$$g - \frac{1}{4}v_1 - \frac{1}{4}v_2 = 7 \tag{85}$$

From this system, it is easy to see that

$$A = \begin{bmatrix} 1 & \frac{1}{2} & -\frac{1}{4} \\ 1 & -\frac{1}{2} & 1 \\ 1 & -\frac{1}{4} & -\frac{1}{4} \end{bmatrix}, b = \begin{bmatrix} 8 \\ 16 \\ 7 \end{bmatrix}. \tag{86}$$

We solve this system for $x = (g, v_1, v_2)$ by applying the right modifications to P. Under u, we obtained $g = 9.2, v_1 = 1.3333, v_2 = 7.4667$

The Optimal Fixed Policy Since we obtained the constituents of the linear system 82, we are ready to apply the policy improvement routine. After obtaining $x = (g, v_1, v_2)$ due to the policy $u = (\mathbf{1}, \mathbf{1}, \mathbf{1})$, we hold values $v_1 = v_1(u), v_2 = v_2(u)$ to later form a new policy

$$u^1 = (c_1^1, c_2^1, c_3^1) \tag{87}$$

and obtain a larger or equal average return $g = g(u^1) \geq g(u)$ under the probability distribution $p_{ij}(c_i^1)$ and return vector $r_{ij}(c_i^1)$. Iterating these steps until $g(u^m)$ converges to a fixed value g^*

Policy		Gain		Policy		Gain		Policy		Gain
1				**1**				**2**		
1	$\rightarrow$	9.2	$\rightarrow$	**2**	$\rightarrow$	13.1515	$\rightarrow$	**2**	$\rightarrow$	13.3445*
1				**2**				**2**		

Figure 14: *The policy improvement routine applied to the taxicab problem, starting with the policy that tells the driver to cruise in every town to find a passenger.*

will put out the optimal fixed policy on our taxicab problem. Figure 14 shows the progression in the policy space for the taxicab problem, as well as the outcomes obtained.

Tire Manufacturing [1]

Assume we are in charge of a machine that is used to produce tires. The machine has two bladders inside, to produce two tires independently. Anytime we want to produce new tires, there is a likelihood of the bladders inside the machine to fail, leading to produce junk, and also bladder itself going obsolete. After each tire produced with one bladder, that bladder is assumed to have a higher likelihood of failure.

When a bladder fails, it has to be changed so that production can continue as usual. Of course, to solve this problem, one has to stop the machine and let the mechanic fix it, and all these actions come with implicit costs:

1. The cost of a new bladder is c_1.
2. The cost of a scrap tire to the company is c_2.
3. To change a bladder, the machine has to be stopped and repaired by a mechanic. The mechanic has a fee for fixing the machine and replacing bladders, c_3.
4. Additionally, while the machine is open to be fixed in operational hours, no tire is produced. Hence, there is a cost of lost production time, c_4.

We consider calling the mechanic outside the operational hours for changing bladders before they fail. By doing this, the time cost does not occur due to being in production only on operational hours, and additionally there will be no scrap tire to get rid of. Hence, we get alleviated from paying the costs c_2 and c_4 if we choose to do this.

Say we got an order of N tires, and in our machine there is two bladders, which have been used i and j many tires before. Let $f_N(i,j)$ be the optimal expected cost of producing N tires,starting with the initial state (i,j), i.e. i tires have been produced by the first bladder, and j tires by second bladder. Then we can calculate $f_N(i,j)$ by seperating the initial stage and iterating over previously calculated costs, $f_{N-1}(\tilde{i},\tilde{j})$ and $f_{N-2}(i^*,j^*)$ for potentially different

$(\tilde{i}, \tilde{j})$, (i^*, j^*). The reason both $f_{N-1}(\tilde{i}, \tilde{j})$ and $f_{N-2}(i^*, j^*)$ will appear in the calculations is due to the likelihood of producing one tire in case of one bladder failing.

The limits of i and j are decided by the probability of success vector that's given or estimated. In this example, it is assumed that after producing 6 tires, a bladder is not viable for use anymore. Our main goal is to reduce the average cost of producing N tires starting with bladders which have produced i and j many tires.

At every stage of production, we can choose to apply the controls below:

1. Change both of the bladders at the same time outside operation hours (Replace Both)
2. Change the first bladder (Replace First)
3. Change the second bladder (Replace Second)
4. Try to produce two tires (Produce Both)

After choosing the controls that results in the minimum cost, we will have the total cost $f_N(i, j)$ with initial state (i, j) to produce N tires.

To see the results of the controls, we will see the expected outcomes of the productions due to each control that's chosen for states (i, j), $i, j = 0, 1, 2, \ldots, \texttt{lifetime}$, and quantify the total cost of each control, to later choose whichever gives the minimum. For example, the state $(0, 0)$ will have the control element "Produce Both(PB)" chosen at all stages (except the first, explained later), simply due to the fact that it is guaranteed to produce two tires without having any cost associated with the durability of the production machine.

To start the calculations, we write the recursive cost function under each control element:

1. RB(Replacing Both Bladders): Two bladder cost will occur, and also the mechanic will ask for a fee for opening the machine. These changes will be done outside the operational hours, so there will be no cost associated to downtime. Additionally, there will be no scrap to get rid of. Yet, there will be no tires produced. Hence, there will be the same number of produced tires with new bladders. Hence, the total cost will now be $2c_1 + c_3 + f_N(0, 0)$
2. RF(Replacing First Bladder): One bladder will be changed, and mechanic will ask for a fee. Similar to the previous control, there will be no tires to be scrapped and no lost time. Again, no tire will be produced, but one of the states will convert to zero. Hence, the total cost will be $c_1 + c_3 + f_N(0, j)$ when we are at state (i, j).
3. RS(Replacing Second Bladder): Exactly the same scenario with the previous control, only for the second bladder. Hence, the total cost will be $c_1 + c_3 + f_N(i, 0)$when we are at state (i, j).
4. PB(Produce Both): This control has four terms averaged by the likelihood of success for each outcome. There are four different outcomes with respect to probabilities p_i, p_j when this control is chosen, which will be further investigated below.

Now assume we choose to produce two tires from both bladders on an arbitrary state (i, j).

1. Suppose that we produced two tires. We are at a new stage $N-2$, and both bladders have now produced one more tire, resulting in the state $(i+1, j+1)$. Since producing two tires has a likelihood of $p_i\, p_j$, the weighted outcome out of this case is

$$p_i\, p_j\ f_{N-2}(i+1, j+1). \tag{88}$$

2. WLOG suppose we produce one tire with the bladder of state i, and the other one fails. Then the new state can be chosen from one of two: since the second bladder has failed,it produced scrap and it cannot produce new tires. So, the machine will be repaired and the second bladder will be renewed. In this case, we have a choice on changing the first bladder as well. Since the machine is already stripped down, the cost of time has occurred and the fee for the mechanic has to be paid. If changing the first bladder together with the second bladder is less costly than changing only the second bladder, then we can choose to replace both bladders and reduce the expected cost. Then the total cost in this case will be

$$\min\{c_1 + c_2 + c_3 + c_4 + f_{N-1}(i+1, 0)\,,\ 2c_1 + c_2 + c_3 + c_4 + f_{N-1}(0, 0)\}. \tag{89}$$

   Hence, the term of expected cost of the control PB coming from this case is

$$p_i(1-p_j)\min\{c_1 + c_2 + c_3 + c_4 + f_{N-1}(i+1, 0)\,,\ 2c_1 + c_2 + c_3 + c_4 + f_{N-1}(0, 0)\}. \tag{90}$$

3. In the countercase of 2, i.e. the first bladder failing, the exact same scenario happens with states replaced. Hence, the term of expected cost of the control PB coming from this case is

$$(1-p_i)p_j\min\{c_1 + c_2 + c_3 + c_4 + f_{N-1}(0, j+1)\,,\ 2c_1 + c_2 + c_3 + c_4 + f_{N-1}(0, 0)\}. \tag{91}$$

4. Finally, suppose both bladders failed. Then there is no tire produced, two scraps have to be taken care of, and the machine will have to be repaired, resulting in two bladder costs, 2 scrap costs, repair cost and cost of loss time. On top of this, the stage hasn't changed. The term of expected cost of the control PB coming from this case is

$$(1-p_i)(1-p_j)[2c_1 + 2c_2 + c_3 + c_4 + f_N(0, 0)]. \tag{92}$$

Hence, the control PB(Produce Both) induces an average cost of

$$\begin{aligned} & p_i\, p_j\ f_{N-2}(i+1, j+1) \\ & + p_i(1-p_j)\min\{c_1+c_2+c_3+c_4+f_{N-1}(i+1,0)\ ,\ 2c_1+c_2+c_3+c_4+f_{N-1}(0,0)\} \\ & + (1-p_i)p_j\min\{c_1+c_2+c_3+c_4+f_{N-1}(0,j+1)\ ,\ 2c_1+c_2+c_3+c_4+f_{N-1}(0,0)\} \\ & + (1-p_i)(1-p_j)[2c_1+2c_2+c_3+c_4+f_N(0,0)] \end{aligned} \tag{93}$$

There are two caviats in the future calculations where we have to be meticulous.

1. Since each stage of the calculations require previous two steps, we will have to start setting up values for producing no tires, $N = 0$ and producing one tire, $N = 1$. Producing one tire requires its own controls setup since the control elements given up to this point assumes that both bladders are used, i.e. up to two new tires are produced.
2. The "renewed" states, i.e. at least one of i and j is equal to 0 have to be considered seperately from other states because they are required on the calculations of all the other states. On the guaranteed failure states we are at the boundary of all states, and so there is no $i+1$ or $j+1$ states to take values from in previous stages.

Calculations Figure 15 is the stair representation of the probability of success for the bladders.

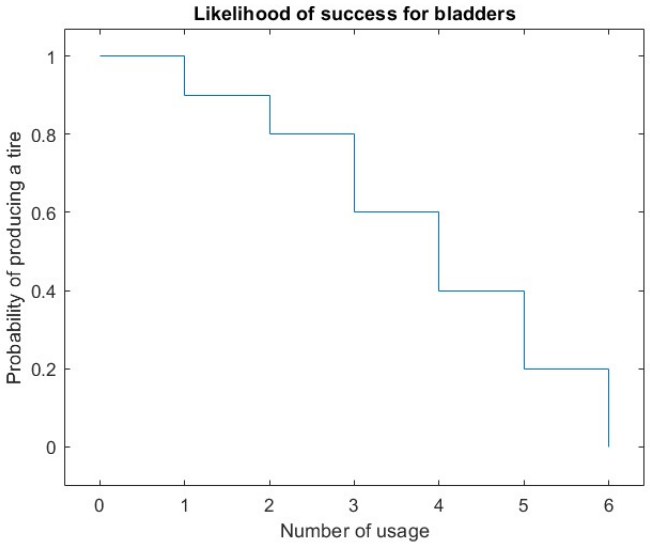

Figure 15: *The probability of success for the bladders to produce a tire, based on how many it has produced*

The bladders are assumed to fail no matter what after producing 6 tires, and so the `lifetime` of a bladder is 6.

Calculating stage n=1 Producing 0 tires has 0 cost. Now, using this fact, we will calculate $f_1(i,j) : i,j = 0,1,2,...,6$, the expected cost of producing one tire at every state.

The controls that were given before are not applicable here because there is no control that results in two tires produced. So we will form special controls for stage 1 in light of the facts below:

1. After producing a tire, the process will be finished. Hence, replacing a bladder with non-zero probability of producing a tire will definitely not be the optimal control.
2. Only one of the bladders will be used, and the other bladder will not be operating, resulting in no change in the number of tires produced with the non-operating bladder.

For this reason, we will have control elements to be applied at each stage as below:

1. Produce With First(PF): The first bladder is used, second bladder is not operated.
2. Produce With Second(PS): The second bladder is used, first bladder is not operated.
3. Replace First Bladder(RF): Replace the first bladder without running the machine and not replacing the second bladder.
4. Replace Second Bladder(RS): Replace the second bladder without running the machine and not replacing the first bladder.

Costs incurred by each of these controls are given below, assuming that we are on state (i, j):

1. Produce With First(PF): First bladder has a probability of success p_i since it has produced i tires. If succeeded, there will be no change in the bladders and the process will be finished, hence creating no cost. If it fails, on the other hand, then we can choose to replace the first bladder or use the second bladder to produce the tire. Choosing to replace the first bladder will guarantee the production of a tire, but at the cost of replacing one bladder with the help of the mechanic. Choosing to produce with the second bladder, which has a probability of success $p_j, j = 0, 1, 2, \ldots,$ `lifetime`, produces a tire, the process will be finished again, creating no costs. Finally, if both bladders fail in sequence, then we will replace only one bladder, and only then we will have a cost for the process. Hence, the cost associated with this control element in stage 1 is $(1 - p_i) \min [c_1 + c_2 + c_3 + c_4, (1 - p_j)(c_1 + 2c_2 + c_3 + c_4)]$.
2. Produce With Second(PS): Same with PF, only the indices i and j are replaced.
3. Replace First Bladder(RF): The repair will be done outside the operational hours, so only the cost of one bladder and the mechanic's fee will occur, next to $f_1(0, j)$ since no tires have been produced. Hence, the cost associated with this control is $c_1 + c_3 + f_1(0, j)$.
4. Replace Second Bladder(RS): Similar to RF, the cost associated with this control is $c_1 + c_3 + f_1(i, 0)$.

After finding an expected cost that is minimal for a specific state (i, j), we store this value and the code of its associated control element in the entries related to the state (i, j). Below

are the tables for optimal expected costs and optimal controls for producing only one tire, for each state $(i, j), i, j = 0, 1, 2, \ldots,$ `lifetime` $= 6$, where rows are the first bladder's number of usage, and columns are for the second:

optimal outcomes	sb:0	sb:1	sb:2	sb:3	sb:4	sb:5	sb:6
fb:0	0	0	0	0	0	0	0
fb:1	0	0.5700	1.1400	2.2800	3.4200	4.5600	5.6000
fb:2	0	1.1400	2.2800	4.5600	6.8400	9.1200	11.2000
fb:3	0	2.2800	4.5600	9.1200	13.6800	18.2400	22.4000
fb:4	0	3.4200	6.8400	13.6800	20.5200	27.3600	33.6000
fb:5	0	4.5600	9.1200	18.2400	27.3600	36.4800	44.8000
fb:6	0	5.6000	11.2000	22.4000	33.6000	44.8000	52.0000

optimal controls	sb:0	sb:1	sb:2	sb:3	sb:4	sb:5	sb:6
fb:0	PE	PE	PE	PE	PE	PE	PE
fb:1	PE	PE	"PE"	"PE"	"PF"	"PE"	"PF"
fb:2	PE	PE	"PE"	"PE"	"PF"	"PE"	"PF"
fb:3	PE	PE	"PE"	"PE"	"PE"	"PE"	"PF"
fb:4	PE	PS	"PS"	"PE"	"PE"	"PE"	"PF"
fb:5	PE	PE	"PE"	"PE"	"PE"	"PE"	"PF"
fb:6	PE	PS	"PS"	"PS"	"PS"	"PS"	"RE"

Calculating Later Stages Recall that with the controls RB(Replace Both Bladders) and RF/RS(Replace First/Second Bladder), we don't produce any tires and at least one of i or$j$ become 0, inducing the cost $f_n(0, j)$ or $f_n(i, 0)$. For this reason, if we are storing the necessary information in matrices, when we are setting up the calculations, first we set up the entries which correspond to at least one bladder being renewed, i.e. the first rows and columns of each matrix. After these initial values are set, all "middle" states, i.e. the states where none of the bladders are new or at the end of its lifetime, are calculated straightforwardly, just by applying the control with the smallest cost. For the `lifetime` states, i.e. for the last rows and columns of matrices, the `lifetime` $+$ `1` states are not well-defined, i.e. not a part of the matrices we store the values in, so the data retrieval is likely to create problems if these states are calculated similarly to "middle" states, i.e. states where indices are not 1 or `lifetime`.

Below are the tables for optimal expected costs and optimal controls for producing fifty tires, for each state $(i, j), i, j = 0, 1, 2, \ldots,$ `lifetime` $= 6$, where rows are the first bladder's number of usage, and columns are for the second:

optimal outcomes	sb:0	sb:1	sb:2	sb:3	sb:4	sb:5	sb:6
fb:0	764.5838	781.6505	794.5768	805.2735	812.1936	816.5207	816.5838
fb:1	781.6505	798.4655	811.1907	821.9238	828.8642	833.3027	833.6505
fb:2	794.5768	811.1907	823.6258	834.0984	841.0586	845.7147	846.5768
fb:3	805.2735	821.9238	834.0984	844.1545	850.6203	855.4366	857.2735
fb:4	812.1936	828.8642	841.0586	850.6203	856.3823	860.5782	864.1936
fb:5	816.5207	833.3027	845.7147	855.4366	860.5782	864.2867	866.5838
fb:6	816.5838	833.6505	846.5768	857.2735	864.1936	866.5838	866.5838

optimal controls	sb:0	sb:1	sb:2	sb:3	sb:4	sb:5	sb:6
fb:0	"PB"	"PB"	"PB"	"PB"	"PB"	"PB"	"RS"
fb:1	"PB"	"PB"	"PB"	"PB"	"PB"	"PB"	"RS"
fb:2	"PB"	"PB"	"PB"	"PB"	"PB"	"PB"	"RS"
fb:3	"PB"	"PB"	"PB"	"PB"	"PB"	"PB"	"RS"
fb:4	"PB"	"PB"	"PB"	"PB"	"PB"	"PB"	"RS"
fb:5	"PB"	"PB"	"PB"	"PB"	"PB"	"PB"	"RB"
fb:6	"RF"	"RF"	"RF"	"RF"	"RF"	"RB"	"RB"

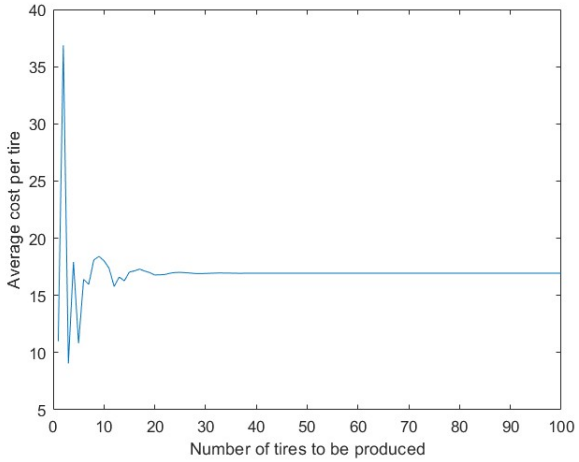

Figure 16: *As the number of produced tires grow, the average cost of producing one tire converges to a finite value.*

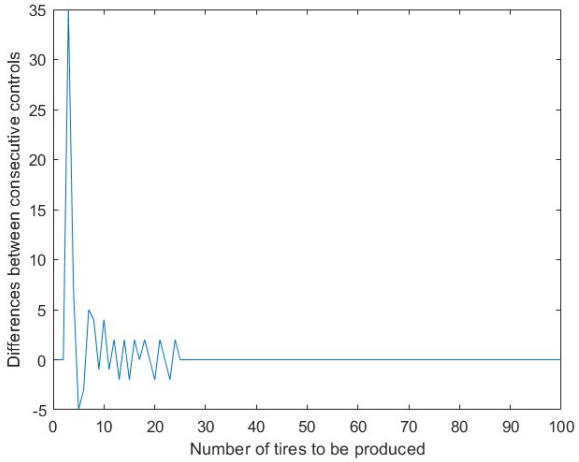

Figure 17: *As the number of produced tires grow, the "differences" between the optimal strategies at each stage vanishes.*

Convergence Results and Average Cost For a Tire For each stage n, we look at the result $\frac{f_n(i,j)}{n}$, which is equal to the average cost of production for one tire with initial state (i, j). The average cost per extra tire converges to 16.94 as $n \to \infty$. As well as that, the policy that's applied at every state also converges, which means the controls we chose for stage 50 is applicable at any stage n larger than 30, as seen in figure 17. The results show how dynamic programming leads to

1. an optimal expected cost of producing a tire, and

2. a fixed policy on each state to be applied to obtain the optimal expected cost.

Conclusion

The problems we worked are targeted by the questions "is there a strategy that guarantees the optimization of processes that require an investment or resource allocation, and if so, how is the theory applied to real world problems?", "how can we maximize the averaged outcomes of non-deterministic processes with the strategies we have?", and "what can we do when analysis techniques are not enough in the investigation of optimal outcomes, especially in discrete settings or functions with uncertainties?". Third question found an answer with the name "dynamic programming". In cargo allocation problems, we found the optimal allocations as an answer to the first question by managing the allocation for each stage one by one by the usage of maximum outcomes from previous stages, instead of having to manage all allocations for each stage at once. In the advertisement campaign problem, we took a deeper delve into the allocation problems by introducing a Lagrange multiplier to the return of the process to help the calculations of best allocations. The taxicab and tire manufacturing problems answered the second question by showing us that fixed policies are the answers that lead to the best outcome. The multidimensional cargo problem was updated with new outcomes, and the numerical values for other problems are obtained as in [1]. The MATLAB scripts are given in the appendix.

The dynamic programming techniques have proven to be highly effective in finding solutions to the five problems given under different settings. A key strength of this approach is its adaptability; the theoretical framework easily accommodates different constraints and policy elements across various problem settings. For future studies, the list of references in the introduction section offers invigorating studies over the foundation set on this thesis.

Preliminary Definitions

- A variable X that represents the outcome of an event, and attains values x from a set S is said to be a *random variable*(r.v. for short) when there is no deterministic way of controlling the value X attains. The set S is called state space, in the sense that each element of S represent the state of the event whose outcome is represented by X. What dictates X to act in the way it acts is called a *probability distribution function* $f : S \to [0, 1]$ obeying the constraint

$$\int_S f(s)ds = 1, \tag{94}$$

which shows the likelihood of elements of S being attained by X, denoted by $P(X = s)$ for any $s \in S$. When the set S contains at most countably many elements, f is called a *probability mass function*, and 94 reduces to

$$\sum_{s \in S} f(s) = 1 \tag{95}$$

for X. For any random variable X, we can calculate $\mathbb{E}[X]$, the average value expected to obtain from X, with the formula

$$\mathbb{E}[X] = \int_S sf(s)ds, \tag{96}$$

where each outcome s of our random variable X is "scaled", or "given the right importance" by the likelihood value $f(s)$ of s.
As a quick example, say X takes values from the set $S = \{1, 2, 3, 4, 5, 6\}$ with a probability distribution $f : \{1, 2, 3, 4, 5, 6\} \to [0, 1]$ where

$$f(1) = f(2) = f(3) = f(4) = f(5) = f(6) = \frac{1}{6}. \tag{97}$$

We can consider X to represent the outcomes of throwing a fair die. f tells us that each outcome from 1 to 6 has equal likelihood of happening. Hence, for n throws of a fair die, we expect to see around $\frac{n}{6}$ many of each number 1 to 6 as the results of throws, and the sum of the outcomes is approximately

$$\begin{aligned} &\frac{n}{6} \cdot 1 + \frac{n}{6} \cdot 2 + \frac{n}{6} \cdot 3 + \frac{n}{6} \cdot 4 + \frac{n}{6} \cdot 5 + \frac{n}{6} \cdot 6 = \frac{n}{6}(1 + 2 + 3 + 4 + 5 + 6) \\ &= \frac{n}{6} \cdot 21 = \frac{7n}{2} = n \cdot \frac{7}{2} = n \cdot \mathbb{E}[X] \end{aligned} \tag{98}$$

- A random variable X is said to have *Poisson distribution with expected value* $\lambda \geq 0$ when the state space is $S = \{0, 1, 2, \dots\}$ and for each element s in S the probability that X attains the value s is given by

$$P(s; \lambda) = \frac{\lambda^s}{s!} e^{-\lambda}. \tag{99}$$

As given in the name, the mean of the outputs of X is equal to λ.

Poisson distribution is commonly used for representing events that occur repeatedly in a fixed amount of time. For example,say a prokaryote organism has a mitosis around every 30 minutes. Then the event of mitosis can be represented by $X \sim Pois(\lambda) = Pois(1)$ where

$$P(k;\lambda) = P(k;1) = \frac{1^k}{k!}e^{-1} \tag{100}$$

shows the likelihood of the prokaryote organism having k mitoses in 30 minutes. In general, by keeping the time interval consistent in each representation, for any positive factor r, the mitosis event can be represented by $X \sim Pois(r \cdot \lambda) = Pois(r)$ with

$$P(k;r\lambda) = P(k;r) = \frac{r^k}{k!}e^{-r} \tag{101}$$

showing the likelihood of the prokaryote organism having k mitoses in $r \cdot 30$ minutes. Letting $r = 2$ means that the organism has two mitoses every $2 \cdot 30 = 60$ minutes, i.e. every hour.

- A sequence $X_1, X_2, \ldots$ of random variables is called a Markov chain if the value X_{k+1} obtains is affected only by the value X_k has obtained;

$$P(X_{k+1} = x_{k+1}|X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k) = P(X_{k+1} = x_{k+1}|X_k = x_k). \tag{102}$$

For example, let's consider a taxicab's daily operations. Letting $S = \{1, 2, \ldots, N\}$ represent a list of towns, one could set up a sequence $X_1, X_2, \ldots, X_n$ that represents the travels of the taxi driver on a specific day. For any $k = 1, 2, \ldots, n-1$, the next destination of the taxi driver is going to be decided by the passenger who is in town X_k. Hence, the places $X_1, X_2, \ldots, X_{k-1}$ are not important for X_{k+1}.

References

[1] Richard Bellman and Stuart E. Dreyfus. *Applied dynamic programming*. eng. Princeton Legacy Library. Princeton, N.J: Princeton University Press, 1962. ISBN: 9781400874651.

[2] Richard Bellman. "On the Theory of Dynamic Programming". In: *Proceedings of the National Academy of Sciences* 38.8 (1952), pp. 716–719. DOI: 10.1073/pnas.38.8.716. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.38.8.716. URL: https://www.pnas.org/doi/abs/10.1073/pnas.38.8.716.

[3] Richard Bellman. "A Markovian Decision Process". In: *Indiana Univ. Math. J.* 6 (4 1957), pp. 679–684. ISSN: 0022-2518.

[4] Richard Bellman and Robert Kalaba. "DYNAMIC PROGRAMMING AND STATISTICAL COMMUNICATION THEORY". In: *Proceedings of the National Academy of Sciences* 43.8 (1957), pp. 749–751. DOI: 10.1073/pnas.43.8.749. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.43.8.749. URL: https://www.pnas.org/doi/abs/10.1073/pnas.43.8.749.

[5] Richard Bellman and E Stanley Lee. "Functional equations in dynamic programming". In: *Aequationes mathematicae* 17.1 (1978), pp. 1–18.

[6] Dimitri Bertsekas. *Abstract dynamic programming*. Athena Scientific, 2022.

[7] Eugene A Feinberg and Adam Shwartz. *Handbook of Markov decision processes: methods and applications*. Vol. 40. Springer Science & Business Media, 2012.

[8] Lodewijk Kallenberg. "Markov decision processes". In: *Lecture Notes. University of Leiden* 428 (2011).

[9] W.B. Powell et al. "Approximate dynamic programming for high dimensional resource allocation problems". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 5. 2005, 2989–2994 vol. 5. DOI: 10.1109/IJCNN.2005.1556401.

[10] Martin Messinger and Martin L. Shooman. "Techniques for Optimum Spares Allocation: A Tutorial Review". In: *IEEE Transactions on Reliability* R-19.4 (1970), pp. 156–166. DOI: 10.1109/TR.1970.5216436.