


1999

Numerical optimization of recursive systems of equations with an application to optimal swine genetic selection

Richard Edmund Hawkins
Iowa State University

Follow this and additional works at: <https://lib.dr.iastate.edu/rtd>

 Part of the [Agricultural Economics Commons](#), [Animal Sciences Commons](#), [Biostatistics Commons](#), [Economics Commons](#), and the [Genetics Commons](#)

Recommended Citation

Hawkins, Richard Edmund, "Numerical optimization of recursive systems of equations with an application to optimal swine genetic selection " (1999). *Retrospective Theses and Dissertations*. 12458.
<https://lib.dr.iastate.edu/rtd/12458>

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI[®]
800-521-0600

**Numerical optimization of recursive systems of equations
with an application to optimal swine genetic selection**

by

Richard Edmund Hawkins

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Majors: Economics; Statistics

Major Professors: Jack C.M. Dekkers, James B. Kliebenstein, and Wolfgang Kliemann

Iowa State University

Ames, Iowa

1999

Copyright © Richard Edmund Hawkins, 1999. All rights reserved.

UMI Number: 9950094

Copyright 1999 by
Hawkins, Richard Edmund

All rights reserved.

UMI[®]

UMI Microform 9950094

Copyright 2000 by Bell & Howell Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of
Richard Edmund Hawkins
has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

~~Co~~-major Professors

Signature was redacted for privacy.

~~Co~~-major Professors

Signature was redacted for privacy.

~~Co~~-major Professors

Signature was redacted for privacy.

~~For~~ the Co-major Program

Signature was redacted for privacy.

For the Co-major Program

Signature was redacted for privacy.

For the Graduate College

TABLE OF CONTENTS

ABSTRACT	x
CHAPTER 1 INTRODUCTION	1
1.1 The problem	3
1.2 The swine commercial and breeding industry	5
1.3 Economic model	5
1.3.1 Analytic solutions	6
1.3.2 Computational solutions	7
1.4 Objectives	7
CHAPTER 2 GENETICS	9
2.1 Structure of the model	9
2.2 Some necessary breeding terms and concepts	9
2.2.1 Truncation	9
2.2.2 Mass selection	10
2.2.3 Genotypic selection	10
2.2.4 Disequilibrium	10
2.2.5 Polygenic variance	10
2.3 The breeding value and total genetic value	11
CHAPTER 3 DYNAMIC PROGRAMMING	14
3.1 Introduction	14
3.2 The class of problems	14
3.3 A brief history of dynamic programming	15
3.4 A general model	16
3.4.1 The starting solution	16
3.4.2 Allowed states and choice spaces	17
3.4.3 Contribution to total objective function	17

3.4.4	Making the infinite horizon finite	19
3.4.4.1	Discounted value below machine precision or other tolerance	19
3.4.4.2	Repeated states	19
3.4.4.3	Optimization costs exceed benefits	19
3.5	Bubbles in n-space	20
3.5.1	Missing the bubbles	23
3.5.2	A structure for the bubblettes	23
3.5.3	A structure for the bubbles	26
3.6	The optimal genetic improvement model	27
3.6.1	Replacing choice variables with state variables	27
3.6.2	Solving the model	29
3.6.3	Extending the notion of “neighbor”	31
3.6.4	Collapsing the bubbles	33
3.6.5	Adding discounting	34
3.6.6	An infinite horizon	35
3.6.7	Using default choices	36
3.6.8	Choice costs	36
3.6.9	Changing the horizon	36
3.7	Higher dimensions	37
3.7.1	Change in bubble and bubblette structure	37
3.7.2	Searching the multi-dimensional space	39
3.7.3	Separable substates of the states	40
CHAPTER 4 THE GENETIC PROBLEM IN ONE DIMENSION		41
4.1	Formulation	41
4.1.1	Simple case: one locus	41
4.1.2	Simplified finite number of generations	42
4.1.3	Discounted finite generations	43
4.1.4	Infinite horizon	45
4.2	Optimal control methods for soluble cases	46
4.3	Infinite horizon with testing costs	51
4.4	The breeding problem and dynamic programming	52
4.5	An initial algorithm	53

4.5.1	Simple dynamic programming algorithm	54
4.5.2	Solving the model	56
4.5.3	Mechanics of conversion to frequency space	58
4.5.4	Adding discounting	62
4.5.5	Using mass selection as a default choice	62
4.5.6	The value of mass selection	63
4.5.7	Testing costs	63
4.5.8	Changing the horizon	64
CHAPTER 5 THE GENETIC PROBLEM WITH DISEQUILIBRIUM		65
5.1	Gametic phase disequilibrium	65
5.2	Finding the state variables	66
5.2.1	The state	69
5.2.2	Finding the choice variables	70
5.2.3	Mass selection with gametic phase disequilibrium	70
5.2.4	Translation from state to choice variables	70
5.2.5	Bounds for the fractions that can be selected	72
5.2.6	Changes in bubble structure from the one-dimensional model	73
5.2.7	Plateaus in the state space	74
5.2.8	Ridges in the state space.	74
5.2.9	Discounting and changing horizon in two-dimensions	75
CHAPTER 6 FOUR DIMENSIONS: THE GENETIC PROBLEM WITH DIFFERENTIAL SELECTION		76
6.1	Changing the model	77
6.2	Changing the pattern of search	79
6.3	Adding a cache	79
6.4	Partial additivity	80
CHAPTER 7 OPTIMAL BREEDING WITH THE ESTROGEN RECEPTOR LOCUS		81
7.1	Estimated breeding value and the ESR	84
7.2	Choosing which animals to breed with truncation selection	84
7.3	Assumptions and parameter values	86

7.4	Modification of the algorithm	86
7.5	Results	86
CHAPTER 8 SUMMARY AND CONCLUSIONS		89
8.1	Analytic methods	89
8.2	Dynamic programming and bubbles	89
8.3	The cache used for the bubbles	91
8.4	The ESR gene	92
8.5	Conclusions for the dynamic programming method	93
8.6	Conclusions for the discounted genetic model	93
8.7	Future research	94
APPENDIX A GLOSSARY		95
APPENDIX B VARIABLE NAMES AND DEFINITIONS		98
APPENDIX C SELECTED BREEDING PROGRAMS		100
BIBLIOGRAPHY		107

LIST OF FIGURES

Figure 2.1	Three distributions with the same variance	13
Figure 3.1	The choice made from choice space C_t at stage t determines the choice space available in period $t + 1$.}	18
Figure 3.2	Each book in a stack makes a cumulative contribution to the height, but surface area is transitory, determined only by the topmost book.	18
Figure 3.3	Initial trial solution for two stages.	21
Figure 3.4	Partition of space near trial solution into disjoint bubbles.	21
Figure 3.5	Division of bubbles into bubblettes.	22
Figure 3.6	With the bubbles centered around the initial solution shown by the dashed arrow, the solid arrow shows a potential new solution or trajectory through the search space.	22
Figure 3.7	New disjoint bubbles and bubblettes are created, at a finer grain and centered about the best solution from the previous iteration.	24
Figure 3.8	Two types of bubble “miss.”	24
Figure 3.9	Placement of bubbles on grid in state space	28
Figure 3.10	Flowchart for <code>bestVal()</code>	30
Figure 3.11	Redfining the notion of “neighbor.”	32
Figure 3.12	Searching for a bubble with <code>bubbletteAt()</code>	38
Figure C.1	Dekkers’ fifteen generation results for one state variable	101
Figure C.2	The algorithm’s fifteen generation results for one state variable without discounting	101
Figure C.3	The algorithm’s fifteen generation results for one state variable with discounting	102
Figure C.4	The algorithm’s fifteen generation results for one state variable with an infinite horizon	102

Figure C.5	The algorithm's fifteen generation results for one state variable with test costs .	103
Figure C.6	Dekkers' eight generation two dimensional results.	103
Figure C.7	The algorithm's eight generation two dimensional results	103
Figure C.8	The algorithm's ten generation two dimensional results	104
Figure C.9	Dekker's five generation results for four state variables	104
Figure C.10	The algorithm's five generation results for four state variables	104
Figure C.11	Sample output in which the solution was flushed from the cache	104
Figure C.12	The algorithm's five generation results for the ESR without discounting	105
Figure C.13	The algorithm's five generation results for the ESR with discounting	105
Figure C.14	The algorithm's five generation results for the ESR with an infinite horizon	106
Figure C.15	The algorithm's five generation results for the ESR with a differential interest rate so as to estimate elasticity	106

LIST OF TABLES

Table 7.1	Estimated parameters of the ESR gene	87
Table 7.2	Results from optimal breeding of the ESR gene	87

ABSTRACT

A new dynamic programming method is developed for numerical optimization of recursive systems of equations, in which choice variables are continuous, and the choices made determine the allowed choices in subsequent stages of the problem. The method works by dynamically creating bubbles, or subsets, of the total search space, allowing the indexing of states visited for later use, and taking advantage of the fact that states adjacent to a visited state are likely to be visited. The method thereby allows the search of parameter spaces far larger than would traditionally be permitted by computer memory limitations. The method allows an infinite planning horizon, and tests at each stage to determine whether further optimization is worth the costs, reverting to a default choice when optimization is no longer profitable. The method is applied to the quantitative genetics problem of finding the optimal selection choices for quantitative traits using an identified gene and the present discounted value of all generations. The method is then applied to the Estrogen Receptor Gene (ESR) in swine to find the economic value of genetic testing for this particular gene.

CHAPTER 1 INTRODUCTION

Recent years have seen rapid progress in computational technology, genetics, and the animal breeding industry. While computer speed and storage have increased, advances in molecular genetics have made it possible to test individual animals for the presence of specific genes, while at the same time, the breeding industry has become more concentrated. These advances can be combined to find more efficient methods of improving genetic progress within livestock breeding herds. Particularly, refinements of the long-established method of dynamic programming to search disjoint subspaces allow the use of genetic testing for individual genes to maximize genetic progress, even when such maximization is analytically impossible.

Within living memory, swine were born, raised, lived, and slaughtered primarily on the family farm. They were mostly bred with the farmer's own herd, or with an impressive hog from another nearby farmer. Slaughter would take place on the farm, where the assorted parts of the carcass would be preserved for personal consumption.

At the turn of the century, the majority of hogs were sent by rail to a few regional slaughter and packing houses to feed the growing population of the cities [Vertical Coordination, p. 4-5]. Rather than feeding the family, raising hogs for market was a way to begin or expand a farm with relatively small amounts of capital. Still, though, a herd could be maintained and improved by avoiding obvious inbreeding and breeding the sows to the sires with superior "phenotypic" characteristics, the outwardly observable traits.

The world has changed since then, and hog production with it. Increasingly, additional traits of the hog were seen to have a genetic basis. Some were to be avoided by careful breeding and slaughter, while others were to be sought out. Research has found better ways to breed the hog, and the predominant family operation of farrow to finish is being replaced by a system with separate operations for farrowing, nurseries, and finishing [Vertical Coordination, p. 4]. For some, it became profitable to buy feeder pigs to raise for market rather than breeding their own, while for others it was profitable to raise and sell feeder pigs.

While the raising of commercial swine remained primarily a family operation, the production of breeding stock rapidly became a concentrated industry. Today the market is dominated by breeding companies, each aggressively marketing their own brand of breeding pig. These companies are constantly positioning for market share, and can either command a better price, take a larger share, or both, by a relatively modest genetic improvement. While a tenth of a percent of additional meat per hog may not have been noticeable to a farmer early in the century, today's commercial farmers see an effect similar to Rockefeller's reducing the number of tacks used in barrels: a fraction of a penny per hog can add up. Just as Rockefeller saved \$60, 000 per year by reducing the number of tacks per barrel by one, commercial swine operations can potentially reap a measurable profit by the slightest improvement—and suffer measurable losses for the slightest defect.

The market has changed further since then. While artificial insemination of swine was not practical even ten years ago, today an increasing number of hogs are conceived this way [Lawrence 1600]. While sales in the past were based merely on animal weight, today price reflects carcass quality considerations. Statistical sampling and proxy measures such as the thickness of fat on the back of a hog create price premiums, whereas excess fat will bring a penalty. In a multi-stage operation it is now necessary for each manager to be able to assess the quality of both inputs and outputs, creating further competitive pressure for the industries.

While genetics have been indirectly recognized as affecting the performance of animals long before Mendel's age, resulting in selective breeding, the development of modern genetics and molecular biology has meant that an increasing number of genes that affect quality in various manners have been identified. The estrogen receptor gene (evaluated in Chapter 7) is known to increase litter size [Rothschild 99], while a stress gene causing fainting and reduced meat quality has also been identified [Eikelenboom]. Not only are these genes recognized, but an individual animal may be tested for the presence or absence of the gene *prior* to breeding to avoid less desirable progeny.

While the famous example of Mendel's Peas concerned a *qualitative trait*, or one that is either present or not present, today most genes of interest in livestock genetics concern *quantitative traits*, or those which take a range of values, such as animal size. Rather than fully governing outcome, a gene may be one of thousands which, along with the environment, contributes to the size of an animal, some with relatively *minor individual effect*, and others with a *strong effect*. If these more important genes can be recognized, the possibility exists of identifying a better rule for selecting animals to breed, yielding an increase in quality without an accompanying increase in the regular costs of operation; though there will, of course, be a cost to perform the genetic testing..

Improved selection decisions have a direct economic impact. An improvement in the current generation also improves all subsequent generations—an extra 1% profit from improved genetics in this year’s production raises all future years, which in turn increases the value of the enterprise by 1%. That is, the change is permanent—as is the increase in the value of the enterprise. More importantly, with the improved genetics, the wealth of the operation is improved by a quantifiable amount.

Current breeding methods are based on *phenotypic* information, the observable traits of an animal. From this information, a *breeding value* is estimated, or the statistical expected value of the collective effect of the genes passed by the animal to its progeny [Falconer]. The goal is to maximize rates of genetic improvement, an approach which is closely related methodologically to that of optimizing profits: in fact, it is a subset. In the simplest case, revenue is strictly a multiple of the quantity produced—pounds of milk, for example—and the cost of testing is very low. In these cases, revenue is a simple multiple of the breeding value, production costs are taken as constant, and the optima for genetic improvement are also the economic optima.

More complicated cases can be handled as well. Quality as well as quantity can be handled: leaner pork may yield a higher price per hundredweight. Sales volume may depend upon quality: there is more demand, as well as a higher price, for superior semen for artificial insemination. Finally, there may be a “brand” premium in having fixed a gene as present in the breeder’s animals that exceeds the direct value of the gene. As with revenues, costs may not be constant. It will frequently be the case that testing for the presence of genes in individual animals imposes a significant cost, and that the number of generations to test becomes an additional variable to optimize. The quality of the animal may increase or lower production costs, as well: a sow with larger litters may have increased veterinary costs, partially offsetting the gains, while optimizing for disease resistance could be expected to reduce the costs of raising the breeding herd and commercial animals.

1.1 The problem

While Mendel created a discipline with wrinkled and unwrinkled peas, today’s geneticist faces more complex problems. Creatures have many traits, most of which are influenced by large numbers of genes—enough that they may frequently be treated as having infinite count. In recent years an increasing number of these genes have been identified. Moreover, they act in concert with a large number of as yet undiscovered genes. Given that a gene contributing significantly to a quantitative trait of interest can be detected, it seems likely that this information can be used to improve the herd. Particularly, the best possible improvement *using* the information will be *at least* as good as without the information.

The question is then *how* to use the genetic information. The question as to *how* to use the information is not easy; the first proposed rules, using “genotypic selection,” yielded the result that in the long term genetic improvement *decreases* from the mass selection values [Gibson], those obtained by breeding the animals with the greatest observed values for the trait in question.

If a gene is designated as B when the desirable allele is present, and b when not present, there are three “genotypes,” or types of animals: bb , bB , and BB . A naive approach would be to breed only the BB 's. However, this is far from optimal. Suppose that each B is worth 1, and that the unknown genes yield a standard-normal distribution for the trait. Approximately 5% of the bb 's will draw a value greater than 2 from the distribution, while half of the BB 's will draw a negative number. The 5% of bb 's worth greater than two are clearly more desirable than the BB 's which draw a negative value, which are still worth less than 2 after adding the effects of the BB ; it is therefore desirable to keep some of each [Freund, p. 242].

The question remains, however, as to the optimal combination. By selectively breeding, and with a litter size of 10, a gene can be brought from a frequency of 5% to 99% in about five generations as seen in the breeding programs of Appendix C. This means that if the program were to last for ten generations, with only the last generation of concern, the first five could focus on merely looking for high values from the normal distribution, with the last five focusing on increasing the frequency of the BB gene. This is not the optimal pattern, but is offered to show that the animals without the favored gene remain of value in maximizing the trait in question. Further, a choice must be made as to which BB 's to breed.

While many classes of breeding rules exist, those considered in this study will select animals by *truncation selection*: within each *genotypes*, such as bB : All creatures with an *estimated breeding value* in excess of the truncation point for that genotype are mated randomly amongst all breeding creatures.¹ Further, it will usually be assumed that the herd is arbitrarily large. As such, the mean breeding value in the following generation, as well as all other variables of concern, are degenerate random variables [Davidson, p. 349].

Preliminary work has already been done in this area. Dekkers and van Arendonk considered the case of a single *quantitative trait locus*, or QTL, at which a detectable gene is located, and used optimal control to find the maximum improvement in the final generation of an infinite herd. [Dekkers 98] Dekkers has also solved a similar case with differential selection by gender in unpublished research.

¹Faster progress could be made by selecting mates. However, this would introduce concerns about inbreeding, making the problem far more complicated. It is prudent to first solve the simple problem and then approach the more complicated problem with the knowledge gained.

Two basic scenarios, a small finite number of periods, and an infinite horizon using present (discounted) value, will be used in this research, and each will be examined both with optimal control and numeric methods. In the simplest model, only the genetic improvement over time is considered. This problem is computationally harder, due to stiff Hessian matrices, and is of value in testing the robustness of the algorithms. Furthermore, an analytic solution is known from the work of Dekkers and van Arendonk [Dekkers 98], and allows verification of the algorithm's behavior prior to analysis of cases with no analytic solution.

1.2 The swine commercial and breeding industry

The pork production industry in the United States is becoming increasingly concentrated, and increasingly sensitive to costs. From a mere 7% in 1988, firms producing 50, 000 or more hogs per year marketed 37% of the total output, more than a third, in 1997 [Lawrence 1599]. The same study found that this structural change is likely to continue, with growers in all size categories planning on expansion. The increased supply will put greater pressure on profit margins, and even a slight edge might be the difference between remaining in production or leaving the industry.

Production of breeding animals has become even more concentrated. The commercial swine industry has evolved from breeding females being raised on the farm to being raised in specialized herds and sold to commercial producers or replacement feeders.

1.3 Economic model

The breeding model translates directly to an economic model by incorporating the present value for a finite number of generations. Revenues are the present discounted value of some function, possibly linear, of the trait in question plus any brand premium,

$$\sum_{t=0}^{\infty} \rho^t [R(\bar{A}_t, p_t) + B(\bar{A}_t, p_t)] \quad (1.1)$$

where ρ is the discount factor, the revenue R reflects the value of animals of the grade indicated by the average polygenic breeding value, \bar{A}_t in generation t which depends upon gene frequency p_t , the fraction of the loci in the herd that have the gene, and other factors, and the brand premium B is an amount, possibly zero, paid beyond the revenue at that same level when the gene is fixed, i.e., gene frequency $p_t = 1$.

Costs may be divided into three significant areas: fixed costs, which will include costs relating to the size of the herd, variable costs based on the quality of the herd, and testing costs. As it is assumed

that the size of the herd is already chosen, and that a full herd will be kept, fixed costs may be ignored in all cases, as they are the same. Cost is discounted in the same manner as revenue.

Quality cost is a marginal cost, measuring from the fixed cost, that may be either positive or negative. Disease resistance, for example will reduce veterinary costs, yielding a negative quantity cost. On the other hand, increasing litter size will increase costs, as there are more animals to feed and house. This increased cost, however, should be more than offset by the increased revenue from the extra animals. Similarly, breeding for increased milk yield or animal size may cause the animals to eat more, raising costs while producing additional salable product.

The final type of cost is that of testing for the gene. If testing were to continue forever, it could be treated as a fixed cost of the enterprise. However, this is unlikely: unless testing can be done by a casual visual inspection, such as a different color of animal, it will have at least *some* cost. Once the gains from use of this genetic information, as compared to the program which would be used without the information, exceed the cost of testing, testing will cease. Therefore, the breeding cost is a variable cost that is expected to change permanently to zero at some point in the future.

Combining these revenue and cost streams yields the total value of the breeding enterprise,

$$\sum_{t=0}^{\infty} \rho^t [R(\bar{A}_t) + B(\bar{A}_t, p_t) - F_t - Q(\bar{A}_t, p_t) - T(p_t, t)] \quad (1.2)$$

which is to be maximized, where ρ is the discount factor, F_t is the fixed cost in that time period, Q the quality cost, and T the cost of testing. Only models in which T depends solely upon p_t until testing is halted, after which it becomes 0, will be considered; strategies such as testing alternate generations will be ignored.

Very little consideration will be given to purely genetic models without economic consequence. In fact, there is only one possible genetic question that can be answered, namely “What is the greatest genetic progress possible in N generations?” where N is a fixed number. Any other question must involve a state from at least two generations after the initial state, which means that these must be weighted. Barring Divine Revelation, the question of how to weight generations is inherently a question of relative economic value.

1.3.1 Analytic solutions

In the simplest case, with a finite horizon, Dekkers and van Arendonk have used optimal control theory to show a partial analytic solution, solving recursively from the final period [Dekkers 98]. Analytic solutions were found for the control equations, and iterative solutions applied to produce the

optimal breeding values. The In section 4.2, it is shown that an analytic solution does not exist for even the simplest case with an infinite horizon. While values for the variables which maximize the objective function certainly exist, the equations have one too many variables to solve.

1.3.2 Computational solutions

In the case of a finite number of generations, simple second order methods are inadequate to solve the problem. Unpublished work by this author has found that choices made in the early generations have very little weight in the final generation, and the Hessian matrix becomes stiff. These could conceivably be solved by augmented Newton methods, a genetic algorithm, or both. However, the various Newton methods require that derivatives of the objective function exist in closed form, which will not always be possible for genetic problems, while a genetic algorithm solving the simplest version of the problem was unacceptably slow, taking several hours.

The infinite horizon presents a simpler problem in some ways. The discounting of the future reduces the stiffness found in the simple model—but will make the “far off” generations, rather than the early generations, nearly irrelevant. The key is in determining how many generations are necessary to be a “large number.” That is, is it fifteen, a hundred, or five hundred generations that must be considered to approximate infinity?

Two broad categories of computational solutions are sought: partial analytic solutions which can be completed with standard numerical methods, and “brute force” methods which can handle methods not at all amenable to analytic solution. The first category is preferable when possible, but would require analytic work for each variation of the problem proposed. However, solutions from these methods can utilize known and established methods to complete problems, and will be certain of achieving optimal solutions. Unfortunately, such partial solutions generally do not exist for problems of economic interest. Accordingly, finding methods for the second class are also desirable, and will be the focus of this dissertation. In some cases, it may be possible to provide a proof of convergence for a class of problems, and in others it may not. However, the methods likely have value even when proof is impossible: while it may not be possible to guarantee optimality, it remains simple to compare the proposed solution to the best *existing* solution—for a breeder, an improvement upon current output is valuable, even if there may be an unknown better improvement.

1.4 Objectives

The objectives for this research are to:

1. develop an approach to quantitatively evaluate genetic improvements in a swine herd.
2. develop an approach to assess the economic value of genetic information on identified genes.
3. provide an application of the program to a specific case of an identified swine gene.

To achieve these objectives, the following sub-objectives must be accomplished:

1. develop a concise formulation of the recursive optimization problem.
2. develop an algorithm or methodology which can find the optimal values for the control variables.
3. provide an implementation of this algorithm for the specific case of maximizing the present discounted value of a herd in which one or more QTL's, or Quantitative Trait Loci, have been identified, taking into consideration the cost of testing. The program code for this algorithm should be as modular as possible to allow for adoption to other problems.

CHAPTER 2 GENETICS

Some standard assumptions are made while working with theoretical genetics. These assumptions will generally apply to very large groups as well. Fundamentally, they are large sample results from the Central Limit Theorem, at such a sample size that variance of the population mean has dropped to zero.

2.1 Structure of the model

Given an initial population in period 0, it is sought to maximize the average phenotypic value P of the animals T generations later, where the phenotypic value is the total effect from genotype g for an identified QTL, polygenic value A , and environment E :

$$\bar{P} = \bar{g} + \bar{A} + \bar{E} \quad (2.1)$$

\bar{E} will be assumed to be 0 in all cases.

2.2 Some necessary breeding terms and concepts

2.2.1 Truncation

Selection is by truncation of the breeding intensity I . For each m , a cutoff point x_{mt} is chosen. Animals with $I_{imt} > x_{mt}$ are selected to breed for the next generation, while the rest do not breed. The breeding animals then randomly choose mates.¹ This cutoff point can also be expressed as a fraction, f_{mt} , which describes the portion of animals of that type bred. Then

$$f_{mt} = 1 - F_{mt}(x_{mt}) \quad (2.2)$$

where F_{mt} is the cumulative distribution function for $I_{f_{mt}}$.

¹Choosing mates for them increases problems with inbreeding of other genes.

2.2.2 Mass selection

In mass selection, $b_{mt} = h^2$ for all m . Thus

$$I = h^2 g_m + h^2 (P - g_m) = h^2 P \quad (2.3)$$

The genotypic information is simply ignored, and the same truncation point is used for each genotype.. This is the simplest selection method.

2.2.3 Genotypic selection

For genotypic selection, $b_{mt} = 1$ for all m . This is an initial attempt to take the genotypes into account. Thus

$$I = 1g_m + h^2 (P - g_m) = (1 - h^2) g_m + h^2 P \quad (2.4)$$

Genotypic selection is the optimal behavior if only a single breeding is to occur.

2.2.4 Disequilibrium

After a selection with different cutoff points for the different genotypes, a negative correlation between the polygenic values and the major genotypes of parents will exist. This is considered at length in Chapter 5.

2.2.5 Polygenic variance

Simple models assume that changes in the mean breeding value are small, and that therefore changes in the variance of the breeding orders are of second order smallness and need not be considered. However, the models considered here attempt to maximize the change that can be made, and this would appear to no longer be a reasonable assumption. The changes in the mean are a function of the variance. If the variance drops significantly due to selection, gains from selection will be overstated, as polygenic improvement is proportional to polygenic variance.

There are two sources of change to the variance of the polygenic distribution. The first is the Bulmer effect [Bulmer, pp. 126-131], which is a reduction in variance due to gametic phase disequilibrium. This effect can be calculated as

$$\sigma_{gt}^2 = p_t (1 - p_t) \quad (2.5)$$

$$\sigma_{pt}^2 = E [P_t^2 - E [P_t]^2] \quad (2.6)$$

$$E [P_t] = p_t^2 E [P_t^2 | m = 2] + 2p_t (1 - p_t) E [P_t^2 | m = 1] + (1 - p_t)^2 E [P_t^2 | m = 0] \quad (2.7)$$

where $E[\cdot]$ is the expectation operator. The total variance can be expressed as

$$\sigma_{At}^2 = \sigma_{gt}^2 + \sigma_{pt}^2 + 2\rho_t\sigma_{gt}\sigma_{pt} \quad (2.8)$$

where the subscripts A , g , and p indicated the breeding, genotypic, and polygenic distributions. While this can be taken further, it suffices for the present to observe that the variance is changing due to selection, and that the simple model does not account for this. Additionally, Bulmer has shown that while variance decreases due to selection, the decrease is asymptotic to a non-zero level in [Bulmer 71]. As such, assuming that such an asymptote has been reached in the existing herds is not a particularly strong assumption. This effect is also considered in [Dekkers 92].

2.3 The breeding value and total genetic value

There are two contributors to the breeding value: the major genes, and the polygenes. As the population is arbitrarily large, each range of values for P is present in its statistically expected value. It is assumed that it is the mean of the phenotypic value for the entire population that is of interest, rather than the traits of individuals. In a large population, the average breeding value and average phenotypic value are equal. For example, it is the milk production of a dairy herd that is most relevant, rather than how much a particular cow produces. The discussion that follows considers only a single locus with two alleles, for the sake of simplicity, and largely follows the development and terminology in [Dekkers 98]. The concepts carry over directly to the case of multiple loci.

Genotypes are tagged by the variable m , taking the values 0, 1, and 2, referring to the number of times the favorable allele is present. g_m refers to the genotypic value of the gene, and takes the values $\{-a, d, a\}$ for $m = \{0, 1, 2\}$. In the case of additive major genes, which will be the primary case considered, the ‘‘dominance’’ d is zero, meaning that each copy of the gene makes an identical contribution. Chapter 7 will consider the asymmetric effects for the Estrogen Receptor Gene, for which the second copy makes a far smaller contribution.

Allowing i to index individual animals, the polygenic breeding value of an animal can be estimated by \hat{A}_{imt} , based upon the phenotype and heritability the heritability of the trait, and not accounting for the major gene. \hat{A}_{imt} is represented as a deviation from \bar{A}_t , and can be estimates as

$$\hat{A}_{imt} = h^2 (P - g_m)$$

where P is the observed phenotypic value of the animal, and h^2 is the *heritability*, or the portion of polygenic value which is passed to the next generation. Letting b_{mt} be a weight for genotype m in

generation t , the animal has a selection value

$$I_{imt} = b_{mt}g_m + \hat{A}_{imt} \quad (2.9)$$

For genotypic selection, $b_{mt} = 1$. Each of the three types has a different mean, but the same variance, as seen in Figure 2.1.

For generation t , the frequency of the major gene is expressed by p_t . A value of 0 would mean that the entire population were homozygotes without the favorable allele, and a value of 1 would mean that the entire population had it twice. With the same number of males and females selected (each animal breeds once), and random mating, there will be p_t^2 homozygotes with the favorable allele twice (BB), $2p_t(1 - p_t)$ heterozygotes (Bb and bB), and $(1 - p_t)^2$ homozygotes without it (bB). For d equal to zero, an assumption that will be kept for simplicity until Chapter 7, the average polygenic breeding value of the population as \bar{A}_t , the average breeding value can be expressed as

$$\bar{G}_t = a(2p_t - 1) + \bar{A}_t \quad (2.10)$$

Also, \bar{A}_{mt} refers to the average polygenic value of animals with major genotype m in generation t .

Combining, the state of the system at time t is fully described by the values $\{\bar{A}_t, p_t, \sigma_{pt}^2, \mu_{pt}\}$ where μ_{pt} and σ_{pt}^2 are the mean and variance of the polygenic distribution at time t . If disequilibrium is considered, then this set must be expanded to account for the fact that the polygenic distribution is different for each of the three genotypes, and the set becomes $\{\bar{A}_t, p_t, \sigma_{mpt}^2, \mu_{mpt}\}$. Finally, if optimization is being done with a finite time horizon, the number of remaining generations becomes important, and $T - t$ is needed as well.

The problem is then to choose values for b_{mt} to maximize \bar{G}_T , which is homomorphic with choosing x_{mt} or f_{mt} , or alternatively with choosing selection intensity i_{mt} , where

$$i_{mt} = \frac{z_{mt}}{f_{mt}} \quad (2.11)$$

and z_{mt} is the height of the standard normal distribution at x_{mt} [Falconer].

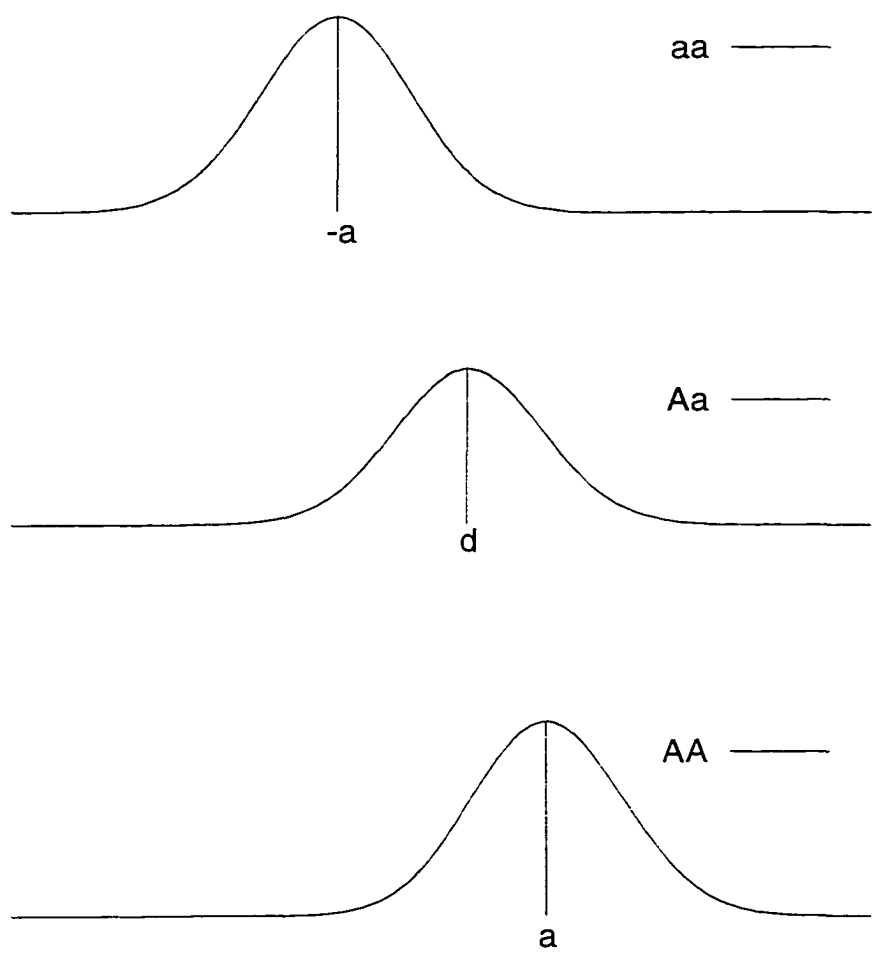


Figure 2.1 Three distributions with the same variance

CHAPTER 3 DYNAMIC PROGRAMMING

3.1 Introduction

The use of dynamic programming in solving problems with continuous variables has generally required that the discretized choice space be compact, or at least contiguous. This sharply limits the allowed dimensions—for example, five $[0, 1]$ variables with a grain of .01 yield 10^{10} potential states, requiring about forty gigabytes of temporary storage for just a single floating point variable for each state.

By using an initial estimate, which need not be close to correct, and forming n -dimensional bubbles around each point in the tentative solution, storage requirements are reduced to a practical level. With this approach, long time horizons, even those approximating an infinite horizon, become practical on modern desktop computers.

An example from selection on a known gene for a quantitative trait in animal breeding is used to illustrate the method.

3.2 The class of problems

The method seeks to solve multiple stage problems in which:

1. The initial state is known,
2. A finite number of state variables exists in each stage,
3. The allowed states in any stage are determined solely by the state in the prior stage,
4. It is possible to specify the contribution of a given stage's state to the total objective function without consideration of prior or subsequent states, and
5. A starting solution is available such that there is a path which is monotonically increasing in the objective function through the state space between this starting point and the optimum.

Also of interest are features that are *not* required with this approach, including

1. Derivatives of the objective function and transition rules.
2. Continuity of the objective function function.
3. A convex search space.
4. A known stage at which to stopping.

It should be noted, however, that while the third is irrelevant to the method, the first two are useful if available, and the method can determine the fourth if it is not known in advance.

3.3 A brief history of dynamic programming

Dynamic programming, in and of itself, is not a new concept. It has been used as a computational method since the 1950's [Bellman], but its ability to solve problems is highly dependent upon available processing power and fast storage (cache or core memory).

Dynamic programming is used in programs that have a fixed number of available states, and in which the "history" of the problem, or the path by which the state was reached, is irrelevant. Dynamic programming can thus be used to solve policy questions, such as the order of applying treatment to fields, in which both *which* steps to take as well as which *order* to take them in are considered [Larson, p. 8]. The value of each state is known, and therefore once both steps of the tentative solution "apply forty tons of phosphor, then plant corn" is calculated, considering the solution, "apply twenty tons of herbicide, then forty tons of phosphor, then plant corn" requires only calculating the effect of the first step, and then using the already calculated value for corn.

While the problems soluble by dynamic programming are inherently discrete, there is a long history of its use in solving continuous problems. This is done by the expedient of discretizing the problem, allowing only certain states, as illustrated in Chapter 5.2 of Bertsekas, among others [Bertsekas]. In some cases, this approximate solution is sufficient. For example, if the problem involves a $[0, 1]$ choice variable, and knowledge of the solution to within .01 is sufficient, then only 100 states need be considered. If such an answer is not precise enough, the first solution can be used as a center for a search over a smaller area with a finer grid, a process known as "successive approximation," a method which is also widely documented [Boudarel, Chapter 4.4]. The wide grid method is named for building an initial wide grid, in multiple dimensions if necessary, about all possible solutions, and using successively smaller grids about each approximation.

However, published numeric solutions almost universally consider compact search spaces (those which are both closed and bounded [Varian, p. 478]). The few examples of non-compact spaces have all used contiguous spaces; a method which is not appropriate for the breeding problem considered in this study. The lack of compactness is not a problem in and of itself; such spaces are used in the method most closely resembling the approach used here, the relaxation method [Boudarel]. This method uses an initial state and trial solution, and allocates storage space around each state on the trajectory. However, this method relies on fixed boundaries for these states, and cannot handle the situation where a choice of state determines the boundaries.

Finally, a broad class of problems can be solved recursively: those which can be reasonably approximated as linear-quadratic systems, in which control is a linear function, and the objective function is quadratic [Chanane], [Pouliot]. The genetic problems of interest don't even come close to linear quadratic—with n generations remaining, the objective is a multinomial of order $2n$ in $2n$ variables, even for the simplest case. Furthermore, the control variables aren't even vaguely linear: among other problems, they involve the inverse of the normal distribution.

3.4 A general model

Before discussing the requirements, it is useful to have a generalized mathematical formulation of the model. It is assumed that the model can be summarized as

$$\pi = \sum_{t=0}^{\infty} \rho(t) V(t, S(t, c_{t-1})) \quad (3.1)$$

where π is the total discounted objective function over all stages, $\rho(t)$ is the discount factor for generation t , V is the objective function, S the state, and c the action chosen. Unless otherwise stated, it will be assumed that the discount factor remains constant,

$$\rho(t) = \rho^t \quad (3.2)$$

and that the objective function for all stages is constant,

$$V(t, S) = V(S_t)$$

3.4.1 The starting solution

All that is required of the starting solution is that it be a legal series of transitions, and that there be a path between this series of actions and the optimal choice which is monotonically increasing in the

value function. While a better starting point will conceivably speed the search, it will only affect the first iteration, as the first iteration reaches the same optimum irrespective of the starting point.

3.4.2 Allowed states and choice spaces

In all cases, there is a choice set C_t at each time stage consisting of the possible transitions from S_t to S_{t+1} , where C_t itself is determined by choices made in prior stages. It is possible to express this set in terms either of choices or successor states as needed, and the two specifications are equivalent. That is, if C_t is the set of choices, or action set, available at stage t , and c_t is the choice actually made, C_t itself is determined by $\{c_s : s < t\}$, as in Figure 3.1

$$C_t = C_t(c_{t-1} | C_{t-1}(c_{t-2} \dots c_0 | C_0)) \quad (3.3)$$

This complicates matters, as a change at time s means that there is not only an interaction between C_t and c_s , which can be handled by considering cross partial derivatives; but that c_s actually *determines* C_t , and thus to speak coherently about effects from changing c_s requires calculating the changes in $\{c_t, C_t : t > s\}$ that result.

Also, it is assumed that the choice sets resulting from alternative choices made in prior stages are not completely disjoint. If this is not the case, then there is no point in using dynamic programming, the strength of which lies in reusing calculated states. Conversely, if the successor choice sets are identical for all choices, or even known in advance, the method herein is unnecessary; its purpose is to adaptively handle subsets of a space too large to search.

3.4.3 Contribution to total objective function

The requirement that the contribution of a given stage to total objective function be known is almost trivial: as calculation of the value of the state entails finding the values of all successor states, the contribution is simply the sum of these states. More generally, it is assumed that there are two components to the objective function, a cumulative portion and a transitory portion. The cumulative portion will remain regardless of future values of the state variables, while the transitory portion depends solely on the current values of the state variables.

For example, suppose a pedestal is to be built by stacking books of different sizes, as in Figure 3.2, and that both the height and the surface area at the top of the pedestal contribute to the quality of the objective function. Each book adds to the height, a cumulative effect; but only the top book determines the surface area, a transitory effect lost when the next book is added. However, the transitory effect

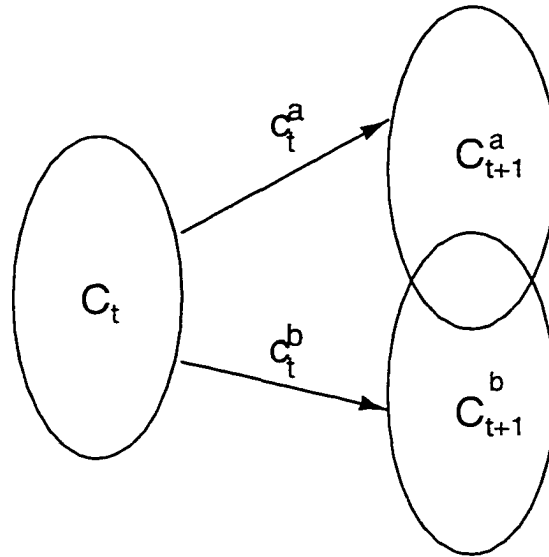


Figure 3.1 The choice made from choice space C_t at stage t determines the choice space available in period $t + 1$.

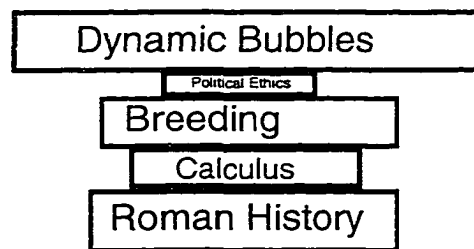


Figure 3.2 Each book in a stack makes a cumulative contribution to the height, but surface area is transitory, determined only by the topmost book.

is not totally irrelevant: the size of any given book determines the possible sizes of books which may become the next level; a book that is far larger than the book immediately below will not balance properly.

In the genetic problem, increases in \bar{A}_t are cumulative; each future generation is increased by the same amount, while increases in p_t are transitory—only the current value of p_t is important.

3.4.4 Making the infinite horizon finite

It has been noted that the method herein can handle an infinite planning horizon. There are three ways in which this can be done.

3.4.4.1 Discounted value below machine precision or other tolerance

This is the simplest and least elegant method. Given that machine precision is finite for floating point math, and assuming that the sum is finite, eventually the present discounted value of a distant stage becomes indistinguishable from zero; and the method can be stopped. A sufficient condition for a finite sum is that there exist a finite N such that for all stages later than N , the increase in the undiscounted value of the stage from the prior stage be less than the discount rate, an assumption that will be maintained throughout this work. Alternatively, a cutoff value which is miniscule relative to the objective function can be used; for example, in a financial model, changes smaller than a penny cannot be measured, and differences in the present value of less than a penny cannot be measured.

3.4.4.2 Repeated states

Another method, available for some classes of problems, is to keep a history of ancestor states. Given the the irrelevance of the path by which a state was reached, any state that either chooses itself directly as a successor state (a steady state), or chooses a successor that ultimately returns to the starting state (a cycle), will do so forever, and indicates the end of the search.

3.4.4.3 Optimization costs exceed benefits

This is the method that will generally be considered in this study. Again assuming a finite present value, if there is a “default” behavior specifiable for the system, and some cost, such as an information cost, for choosing an optimal path, at some point the cost will exceed the gains from testing, and the optimal choice is to switch to the default behavior.

3.5 Bubbles in n-space

It is an assumption of the method that the total search space is so large that it is impossible to allocate resources representing the entire space search, whether or not all states are ultimately considered. Furthermore, it is assumed that the space is large enough that it is not possible to allocate pointers for each possible state, each of which take a single machine word. For graphic simplicity, consider a simple case of three variables in the (0,1) range, for each of which the objective function is an increasing function during any stage. It seems reasonable *a priori* to expect that each of the three will asymptotically approach 1 in any good solution..

With an initial trial solution as in Figure 3.5, and a granularity of .01 for each of the three variables, there are 10^6 possible states, a manageable number. However, the researcher is likely to desire a finer granularity, .0001, or even .000001, requiring 10^{12} or 10^{18} states, respectively.

However, the changes in each variable from stage to stage are typically far larger than the grain: there are large ranges in each variable which are known ahead of time to be unlikely to be reached. Instead, some region, a hypercube in \mathcal{R}^3 around each point of the trial solution is most likely to be reached, as in Figure 3.4. While there is no *a priori* reason to assume a hypercube, this cube can be chosen so as to include any size and shape of region.

If reasonable *a priori* bounds can be placed on the size of these cubes for each step, and this region is divided into ten possible values for each of the three variables, each such bubble contains only 10^3 states, which is manageable for even a large number of generations. Figure 3.5 shows five possible states for each state variable. If the trial solution is “close enough” to the optimal solution for that grain, or if the bubbles are large enough, the solution at each point will be within one of the divisions of the cube, and the dynamic programming approach would then seem to be successive steps with finer grain and smaller bubbles in each stage. For example, if the solution shown by the dashed line in Figure 3.6 is found to be the best, from a prior solution as shown by the solid line, smaller bubbles and bubblettes may be found around this new tentative solution, as in Figure 3.7. To this point, it has been assumed that the each step will land in a known neighborhood, that of the bubble previously associated with that step. However, the dashed line lands in the “wrong” bubble, while the dotted line fails to land within *any* bubble.

This process would be repeated until the grain is sufficiently fine that the solution remains stable, or does not change, as the grain is further reduced.

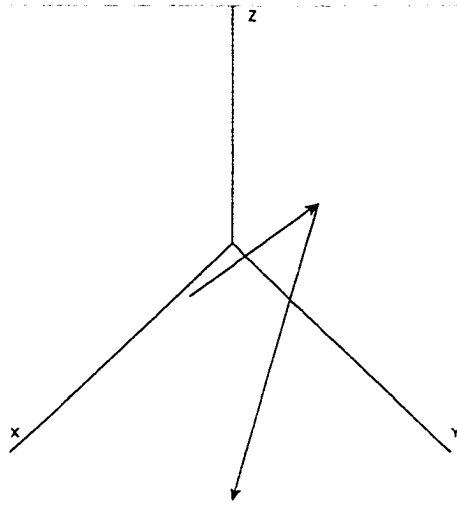


Figure 3.3 Initial trial solution for two stages. The arrows show the progression in three state variables from the initial state to the states comprising the initial trial solution.

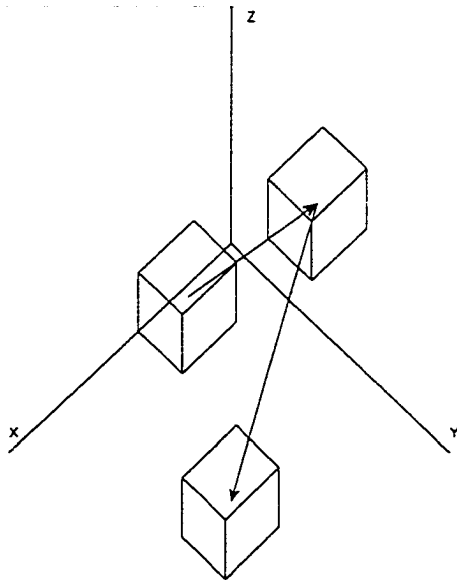


Figure 3.4 Partition of space near trial solution into disjoint bubbles.

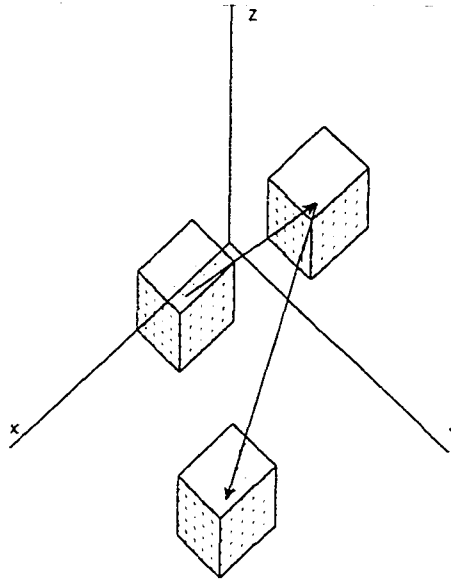


Figure 3.5 Division of bubbles into bubblettes.

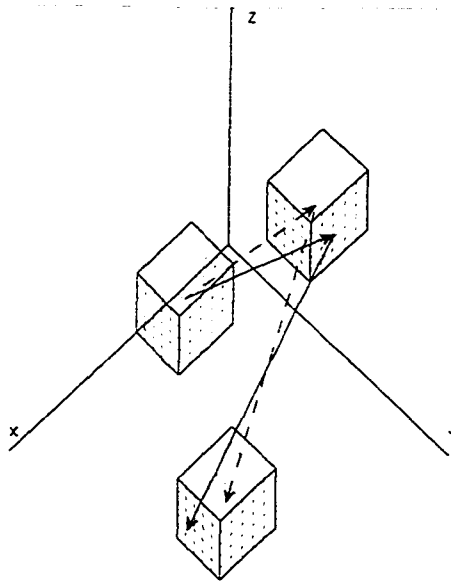


Figure 3.6 With the bubbles centered around the initial solution shown by the dashed arrow, the solid arrow shows a potential new solution or trajectory through the search space.

3.5.1 Missing the bubbles

The foregoing example has a critical implicit assumption: that the number of steps to the solution is known—that is, that the problem is known to have T stages, rather than the optimal choice of T being part of the problem. This is important since the bubbles are identified with a specific step. It is entirely possible that the bubbles for different stages overlap, but this is not a problem. The problem, rather, is that a tentative solution step may step outside of the “next” bubble, as in Figure 3.8. Until now, it has been assumed that each step will land in a known neighborhood, that of the bubble previously associated with that step. However, the dashed line in Figure 3.8 lands in the “wrong” bubble, while the dotted line fails to land within *any* bubble.

This is not a trivial problem, and addressing it is the essence of the method herein. The essence of dynamic programming is to reuse previously calculated states. However, the bubbles are used in this approach precisely because the number of potential states is too great to store, let alone calculate. If a step is made outside a bubble, it would seem likely that the next step is also outside the corresponding bubble. Each of these will require calculations of multiple possible states for comparison with each other. Some method must be found to index or search through the bubbles, so that once a new bubble has been created, later steps can find it and take advantage of its calculations.

Another source of misstep can come with a change in the number of stages in the solution. For example, if the initial solution takes fifteen stages before reaching a state in which testing is not profitable, it may be that a path is found which reaches this level in fourteen generations. This may mean that the thirteenth step proceeds to the bubble previously associated with the fifteenth, as with the dashed line in Figure 3.8.

3.5.2 A structure for the bubblettes

Before turning to the bubbles, it is necessary to consider the nature of the bubblettes.

A bubblette is a fragment of a bubble—a piece representing a single potential state that, with other fragments, can form a bubble in space, as described below. This fragment represents a point in state-space, and all of the information relevant to that state. Particularly, this information must include

1. The location of the state.
2. The next state that should be chosen from the state.
3. The value of stepping to that state, so that it is not necessary to recalculate if the state is considered again.

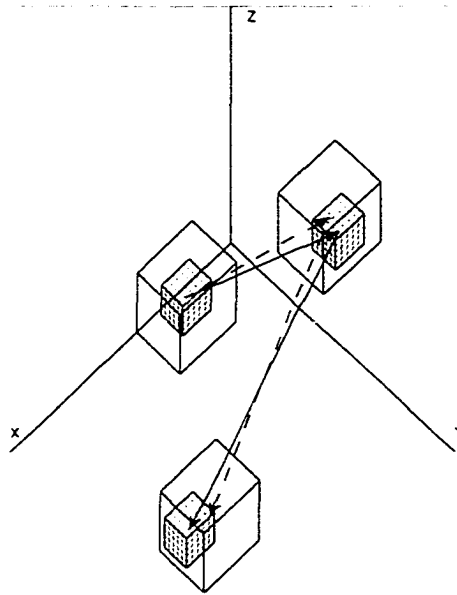


Figure 3.7 New disjoint bubbles and bubblettes are created, at a finer grain and centered about the best solution from the previous iteration.

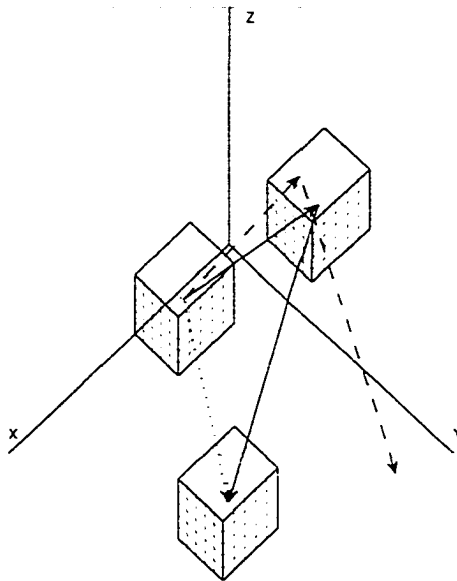


Figure 3.8 Two types of bubble “miss.”
The solution indicated by the dotted line enters a “wrong” bubble, while the dashed line does not enter any existing bubble.

Depending upon the problem, it may also be valuable to include

1. Any other variable, such as the true choice variables (if not the same as the state variables), which are needed or desired in making the transition.
2. Any hints as to what course of action should be taken from prior iterations of the algorithm.

The first will likely be handled by whatever method is used to store and retrieve the bubblette: it is the address of the bubblette. The second and third are the essence of the dynamic programming method: once the act to be taken from this state, and the value of that act is calculated, this data should be available for any future solution that considers this point. The fourth may or may not be necessary, depending upon the problem. For example, in the genetic problem of Chapter 5, the “true” choice variables f_{mt} are information desired by the breeder; they describe the acts to be taken in the real world. On the other hand, an optimal growth problem in macroeconomics may be interested only in the states reached, and not in the underlying variables. Caution should be used in determining which variables to keep, as each variable stored uses valuable memory for each bubblette. Finally, the hints in the fifth may speed execution in some cases, but will not be possible or worthwhile in all instances. For example, after an iteration of the algorithm, it is possible to store the optimal next state for all states considered during the iteration. This can be done by checking each bubblette to see if it was calculated, and multiplying by the relative sizes of the current and replacement grains. However, it will generally be computationally expensive to do so; whether it is worth the cost will vary with the problem. In particular, as the solution stabilizes after several iterations, the best choice will be nearly the same as in the prior iteration; saving this information will eliminate most calculation—at the cost of increasing memory consumed and thereby limiting the number of bubbles available.

Once the contents of a bubblette are specified, it has three possible states of its own: fully calculated, fully uncalculated, and uncalculated with a hint. The fully calculated bubblette must, at a minimum, contain the following information: the fact that it is calculated, all state variables for the bubblette, and the “next” bubblette—the bubblette reached in the next generation. However, more information is desirable, and eases calculation. Particularly, it is desirable to store the choice variables, such as the f_t , which lead to the next bubblette.

If a bubble is not yet calculated, it may yet have a “hint” left from a prior iteration, such as what state the corresponding bubblette in the prior iteration chose, as described above. This hint, perhaps the result for this or an adjacent location in a prior run, can be used as a starting point when the bubblette searches for its values. However, storing the hint does take storage space. As such, it may be

desirable to add a pointer variable to bubblettes for hints, allowing them to point to another bubblette for a hint, rather than storing information itself. This is not possible, however, in the the fully developed algorithm, as bubbles can be destroyed as they grow stale.

3.5.3 A structure for the bubbles

As the bubble is divided into bubblettes, it clearly must, at a minimum, contain storage space for its constituent bubblettes. This, indeed, is the purpose of the bubbles—the optimization will work by checking states adjacent to tentative optima, meaning that once a state is accessed, its neighbors will be accessed with high probability. Thus it is desirable, insofar as possible, to store adjacent states in such a manner that finding one makes finding its neighbors cheap. The search described below is expensive: execution time is cut drastically by reusing the results of a search.

While it is conventional to think of “balls” in n -space for arbitrary regions in multidimensional space, the hypercube is more natural to the computer: a $5 \times 5 \times \dots \times 5$ region is well defined in memory, but the set of all regions within distance 5 is not. The result will be wasted space if a hypersphere was the correct solution, but experience has shown that it will often be necessary to allocate a new bubble next to existing bubbles (indeed, finding such bubbles is the critical portion of the algorithm), and hypercubes “stack” neatly, while hyperspheres can be tangent only at a single point. In other words, hypercubes can be placed next to each other without holes, while other shapes cannot.

In some cases it may be possible to mitigate wasted space by constructing bubbles with arrays of pointers to bubblettes, and only allocating space for the bubblettes when actually used. For eleven possible values for each of five state variables, this means 161051 pointers, which will take 629 kilobytes of storage per bubble on a thirty-two bit machine. While this is a massive amount of storage, and by assumption will be largely unused (it has pointers for all bubblettes within ten five states from the center in every direction, most of which will never be considered), this is still less than ten megabytes for fifteen bubbles, leaving the balance of memory available to allocate for states as they are calculated. However, in the one-dimensional case of Chapter 4, the use of static rather than dynamic arrays was found to approximately double performance.

It will generally be assumed that there are an odd number of levels within for each variable: the center of the bubble comes from some prior knowledge or solution, and a symmetric number of levels on each side results in an odd number.

Merely containing the bubblettes, however, is not enough for the bubble structure. It is necessary to search the bubbles, so that they are not needlessly duplicated. Therefore, bubbles should be of a uniform

size with their centers on a predetermined grid. Figure 3.9 shows possible bubbles and bubblettes in two-dimensions, with grid representations for possible locations of both bubbles and bubblettes.

Another useful feature in a bubble would be reference hints for uncalculated bubblettes. After an iteration of the algorithm, calculated values and choices for the next state will exist for some or all of the bubblettes, many of which will be contained within the corresponding bubbles on the next run. These values can be saved as hints for the corresponding bubblettes within the smaller bubbles of the next pass.

3.6 The optimal genetic improvement model

The initial problem solved considers only the value of the final state; intervening states are ignored. While this is not typical of economic problems, there are three reasons for doing so for the genetic model:

1. There is a reference problem with a known solution for comparison, namely that found by Dekkers, *et al.* [Dekkers 98]
2. It is simpler to test and debug such a model, and
3. The underlying problem is “stiffer, ” and a method that properly solves a problem of this type can be expected to solve easier cases.

The “stiffness” arises from the Hessian matrix. With a reasonably long planning horizon, choices made in the early generations have effects in the final generation which are so small they have little impact upon the final generation.

3.6.1 Replacing choice variables with state variables

One final obstacle must be addressed before attempting a dynamic programming solution: the discretization of the choice space. Most fundamentally, the choice variables cannot be permitted to take continuous values, as dynamic programming relies upon repeated states, and continuous choice variables result in continuous state variables which do not repeat themselves. Discretizing the choice variables does not help with non-linear functions, as the spacing between states resulting from adjacent permitted values of the choice variables will change, and the states will not take the permitted values.

These difficulties suggest that transforming the problem into one of choosing state spaces transitions from those permitted, and afterwards transforming the solution back into the choices that yield the transition, may be the easiest way to solve the problem.

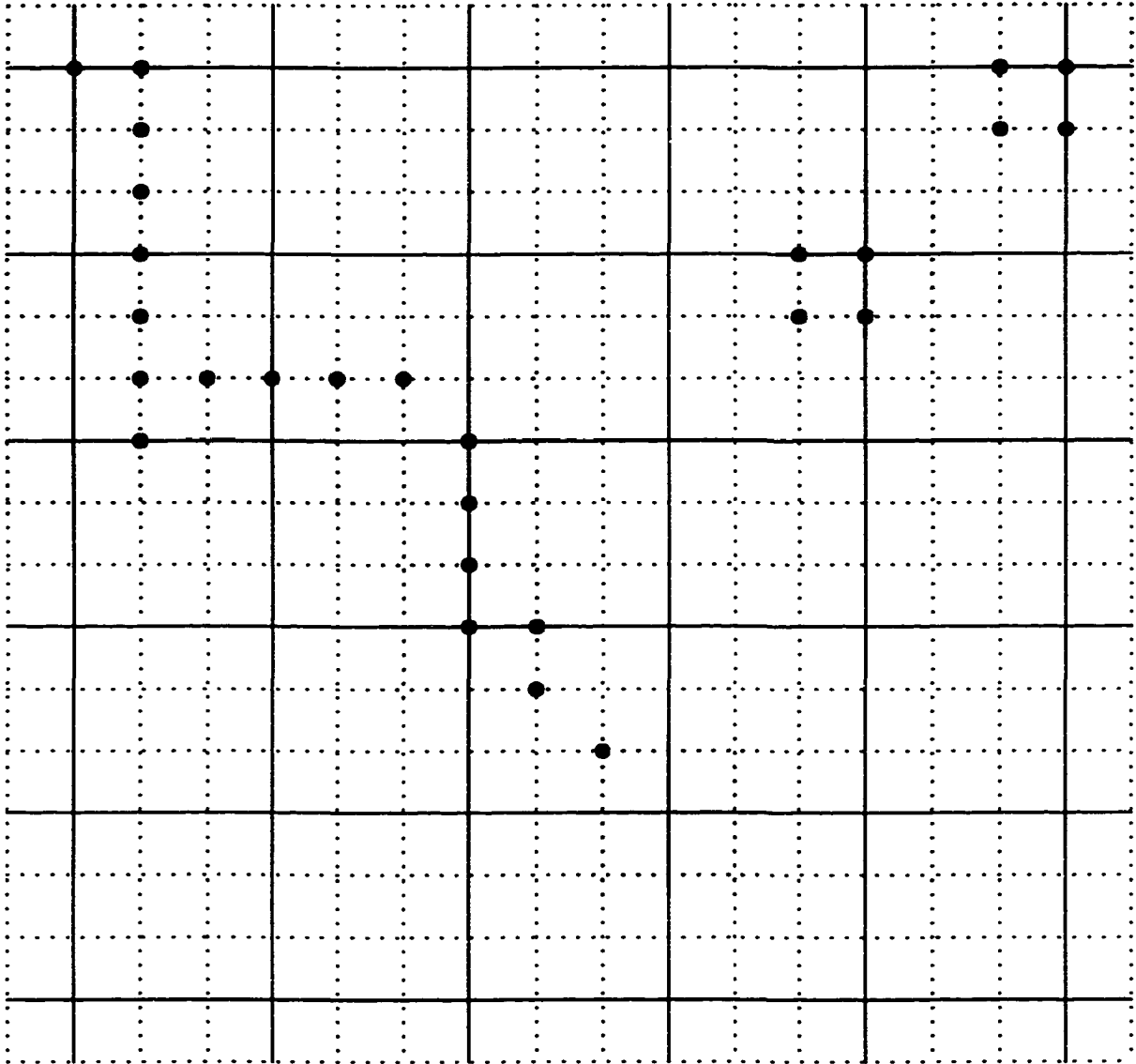


Figure 3.9 Placement of bubbles on grid in state space

Bubbles should be placed on a predetermined grid, as shown by the solid lines. Permissible states are at any of the vertices, and bubblettes actually used are represented by dots. The grey areas represent bubbles, which may or may not have all of their bubblettes used.

3.6.2 Solving the model

Figure 3.10 is a simplified flowchart for a recursive function `bestVal()` which queries a bubblette for its value. When called, `bestVal()` is passed arguments identifying the calling bubble and bubblette, the queried bubble and bubblette, and the stage for which the bubblette is queried.

Beginning in the upper left, the very first task is to see if the bubblette's value has already been calculated. If it has, this value is returned, and the function exits. Assuming that the bubblette has not yet been calculated, a test is made to determine if the bubblette is called for the final stage, which may be "special" for various reasons explained throughout this paper. If it is the final stage, no search is necessary, and the value is calculated, stored, and returned. If the function fails to exit by this point, a search will be necessary. A sequentially issued identification code is created, which will be used to tag bubblettes visited during the search. The purpose is to avoid fully evaluating a bubblette more than once.

It is necessary to identify a bubblette at which to begin the search. Various strategies are possible. The simplest will generally be to use the default choice method. An alternative after the first iteration is to save "hints" from paths calculated but not chosen and to start at the choice that was determined to be the best for the corresponding bubblette of a prior iteration.

The algorithm then bypasses the exit condition and advancement mechanisms in the center-left section of the chart, and proceeds to tagging the base index with the sequential code. Unless another descendant instance of the function looks at the same bubblette, this will prevent later recalculation of the bubblette. Note that for cyclic solutions, a check should be made of the path before issuing this tag. Once the bubblette is marked in this manner, a check is made to see if the transition to this bubblette is possible. If not, a branch is made to check exit conditions. If possible, the function is called recursively to find the value of that bubblette.

The returned value should reflect the transitory contribution of the next bubblette, as well as the discounted value of all future states reached. To this value is added the cumulative contribution made by the step to that bubble.

If the value is the best seen to this point, the base index is changed to point to this bubblette. Ancillary information which is of value later, such as the choice variables that create this solution, may also be stored at this time, depending upon the needs of the particular problem.

In either case, exit conditions are then checked—and will never be met if the index was just changed. While other methods are possible, the neighbors of a bubblette are those for which all indices differ by at most one. While it is possible to write a loop that steps through the multiple dimensions, it is

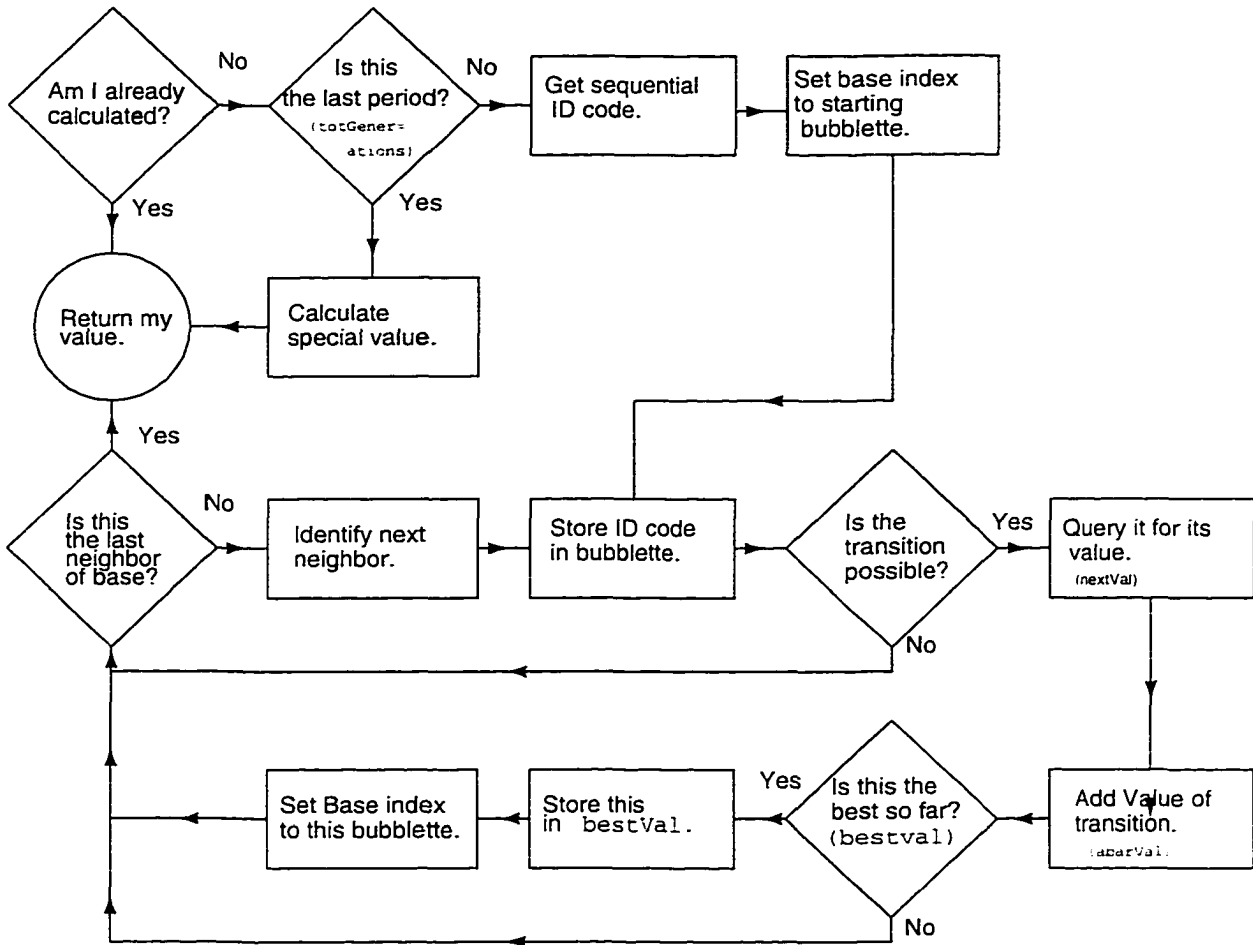


Figure 3.10 Flowchart for bestVal()

bestVal() finds and stores the best action that can be taken for a given state, and recursively calls itself to solve all later stages.

faster to have a constant array indicating all possible relative addresses, allowing a single loop to check all dimensions. If this index reaches its maximal value, then all neighbors of the bubblette have been found either impossible or inferior to the base bubblette, and the value of that bubblette is returned. If not, the next neighbor is identified from the constant array and checked.

Some issues have been glossed over in this explanation. Most importantly, it is assumed that the bubblettes are easily found [see the discussion at page 39 which describes function `bubbletteAt()` during multi-dimensional searches for more information]. Additionally, discounting has been glossed over: the value returned is actually discounted to its present value in the stage of the calling stage.

The diagram is geared to the case of a known *final stage*, and *non-cyclic solutions*. If the stopping time is unknown, rather than returning the best solution found, the best solution is compared to the default choice, which is instead returned if the best solution does not beat the default by at least the information's cost. If cyclic solutions are an issue, as in optimal growth problems, a history should be passed to the function which can be checked as an additional exit condition when a bubblette is revisited.

It is important to note at this time that the bubblette with the highest value will *not* necessarily be chosen. It is not only the value of the bubblette itself that matters, but also the value *added* in stepping to that bubblette. Thus, reaching a bubblette with a value of .3 while gaining .5 from stepping to that bubblette is more valuable than reaching a value of .4 from which a gain from the step of only .2 is achieved. However, the gain of .5 is dependent upon the bubblette from which the step is made: it is entirely possible that the stepping gain is only .1 in reaching that bubblette from a different successor state. As such, the bubblette chosen will vary depending upon the prior state.

3.6.3 Extending the notion of “neighbor”

Generally, it is not difficult to define the neighbors of a bubblette—they are all bubblettes distant by at most one location in each direction. However, this becomes more complex near the boundaries of allowed states. Consider the neighborhood in Figure 3.11: none of the +1 states in the x direction are accessible, due to the boundaries on permissible states. However, the optimum is at C . Simply checking all states adjacent to A brings the incorrect conclusion that A is the optimal point.

Accordingly, the algorithm must consider the nature of the problem and under what situations this problem may arise. In this study, consideration will only be given to the straightforward case of a “superior” and “inferior” state variable, such as p and d in the genetic example of later chapters. In this formulation, the superior variable is not only the primary variable of interest, but is capable of

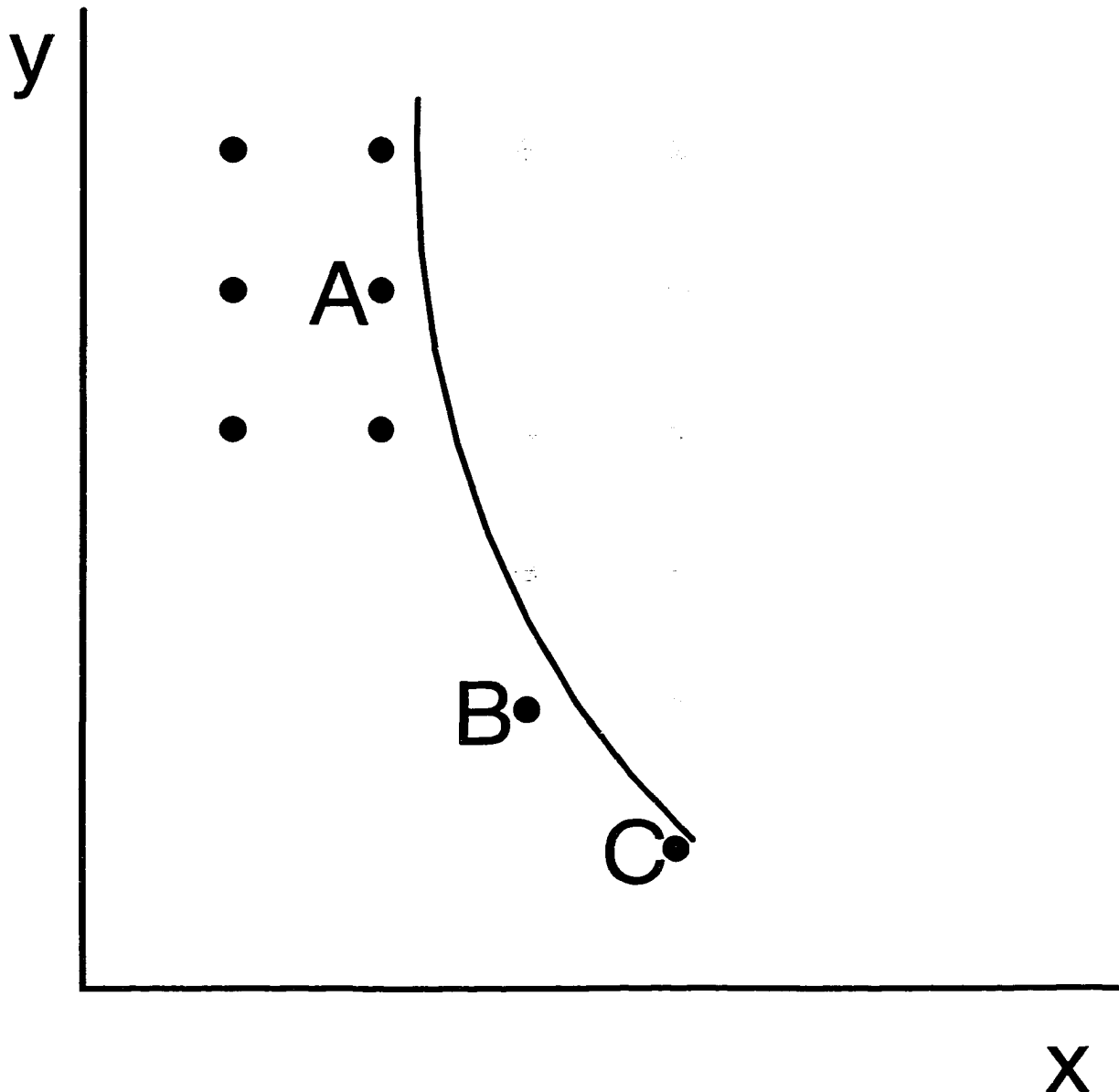


Figure 3.11 Redefining the notion of "neighbor."

At A , none of the $+1$ states in the x direction are accessible, as indicated by their grey coloring. The optimum is at C , while B is superior to A . The line indicates the boundary of permissible state-space. B is therefore located as the nearest state, and checked, allowing the optimum at C to be found.

reaching any state in its range if an appropriate value is chosen for the inferior. The inferior variable may be important in its own right, or it may measure some undesirable side effect, such as d below. For illustrative purposes, it will be assumed that increases in the superior variable are generally desired, although they may be offset by the corresponding change required in the inferior variable.

In Figure 3.11, after attempting all adjacent states, A remained the tentative solution. Before accepting A , however, the algorithm checks to see if any of the $+1$ states for the x direction were accessible and finds that none were. It then detours and calculates the range of possible inferior states for the $+1$ value of x . It then checks the nearest possible state in that direction, namely B . If this point is superior to A , then the search resumes at B . If not, depending upon the nature of the problem, it may be desirable to check additional states before declaring A the optimum. In the two-dimensional breeding problem of Chapter 5, it has been found more efficient to check the two nearest values for the inferior state than to accurately handle floating point to integer conversion and integer rounding so as to correctly locating the “nearest” state. In that problem, due to the shapes of the search space, it is also not necessary to consider decreasing states of the superior variable, but only the increasing states.

3.6.4 Collapsing the bubbles

When the best value for the current grain is reached, the bubbles are “collapsed.” New bubbles are created which are centered about the path chosen, and with the grain reduced. Additionally, hints are stored regarding the “expected” path from bubblette to bubblette. For the center bubblettes, this is simply the next central bubblette for the next stage. For other bubblettes *in the region calculated in the prior iteration*, the hint is the bubblette at the same point in space which was previously reached. Due to the collapsing, there will be somewhat arbitrary choices made as to which bubblettes from the prior iteration correspond to which bubblettes in the new generation—if grain is halved, each possible state in each dimension has three new states to which it could correspond—the same value and the states halfway to the old adjacent states. Rather than spend effort and computation on the matter, this is simply left to the default rounding performed by Fortran; the result will be off by at most a single state in any dimension, and adjacent states will be checked in any event.

A minimum of two bubblettes above and below the center will be created within each bubble. If bubble sizes are flexible, a check is made to determine how many states were actually evaluated in each direction for that bubble, and if larger than two, this quantity is used instead. Furthermore, a check is made to insure that no boundary states were chosen in the final solution—this could indicate that a better state existed after the boundary. In this case, the bubbles are re-centered about the solutions

found, and the iteration repeated with the same grain. This avoids the problem of missing bubbles, and is satisfactory with a single state variable. As a practical matter, flexible bubble sizes seem possible only for a single-dimension, or in cases where bubble misses (stepping to a bubble outside the starting bubbles) are very rare events.

Experience in working with this problem has shown that a bubble size of five, for two states above and below the target, is optimal, at least for the genetic problem of later chapters. In the one-dimensional case of Chapter 4, increasing this to three states above and below the central state increased execution time by approximately twenty percent and did not provide improved results. The algorithm is repeated until the grain of the probability space is satisfactorily small, with the final iteration yielding the reported solution.

3.6.5 Adding discounting

To this point, the only concern has been the maximum amount of *progress* that can be made, and only the final generation has been considered. It is not difficult, however, to modify the model to consider the present value of all generations, and even to allow for the present value of an infinite planning horizon, .

Previously, the program considered only the value of the final stage and added the gain in polygenic value for stage along the way, creating a sum equal to the value in the final generation.

For this discussion, it is assumed that the worth of the state can be divided into a portion that is a permanent and cumulative change and a portion that is not. In the genetic model of Chapter 2, these portions are the polygenic value and the contribution of the major gene, respectively—the polygenic value is the sum of the increases over all generations, which are independent of each other, while the contribution of the major gene depends only upon the value in the current generation.. As such, a change in the cumulative component in stage t increases all generations by the same amount. The discounted value of an increase b , with a discount factor

$$\rho = 1 - r \quad (3.4)$$

is as follows for a finite horizon:

$$\begin{aligned} \sum_{i=0}^T b\rho^i &= \sum_{i=0}^{\infty} b\rho^i - \sum_{i=T+1}^{\infty} b\rho^i \\ &= \sum_{i=0}^{\infty} b\rho^i - \rho^{T+1} \sum_{i=0}^{\infty} b\rho^i \end{aligned}$$

$$= b \frac{1 - \rho^{T+1}}{1 - \rho} \quad (3.5)$$

To calculate a present discounted value of the herd at time zero, then, `bestVal()` need only be modified to apply the identity in [3.5] to the calculated value for the permanent portion of the change in state: discount this value and that returned from `nextVal()`, and add the current value of the major gene. This is accomplished by a simple `if/then` structure in `bestVal()`. The present value of the choice made is reported for each generation, valued at the time the choice is made. This is not the same as the present value of the generation; the value of the current state is not included. Instead, it is the value of the next generation, discounted once, in addition to the values of all future generations discounted appropriately since these too are consequences of the choice made. By calculating in this manner, it is easier to compare the relative value of choices later when choosing whether or not to test.

3.6.6 An infinite horizon

Extending the problem to an infinite horizon is also straightforward and can be accomplished in a number of different ways. The approach which will be used in this study is to use the solution for the n -stage problem as a starting point for the $n + 1$ -stage solution. The starting value for the gene frequency in generation $n + 1$, p_{n+1} , is taken as increasing by half as much from generation n as the increase from generation $n - 1$ to n as in the prior generation but no more than half of the distance to 1.0. That is, the starting value for p_{n+1} is

$$p_{n+1}^0 = \min \left[p_n + \frac{p_n - p_{n-1}}{2}, \frac{1 - p_n}{2} \right] \quad (3.6)$$

By applying standard discount rates, a point is reached where future generations eventually provide very little present value. Furthermore, in choosing the optimal breeding program, it is the *difference* between potential present values of the herd in distant generations that matters. That is, given two possible states for a distant time, it is the difference between the present values of those two states, rather than either state itself, that matters in making a choice. As such, for any arbitrarily small ϵ difference between those states, there exists an n such that the difference between the present values of proceeding for n and for $n + 1$ stages is less than ϵ . This is the first approach taken and requires minimal modifications such that the prior version of the program is placed in a loop which exits when the gain between subsequent stages is less than the specified convergence criterion.

3.6.7 Using default choices

While the above approach could work, it may not be computationally or analytically practical. However, a better method exists. In the simple approach, the entire value of the added stage is an increase, though it is possibly offset by different actions in prior stages (the steps that are optimal for n and $n + 1$ stages are not the same, and thus the combined present value of the first n steps of the $n + 1$ stage solution are worth less than the combined value of the n stage solution). Rather than ignoring the extra stages, it is more efficient to switch to default choices after n . A more efficient solution is to instead of ignoring stages after n to switch to mass selection, and calculate the present value in that manner. This simplifies the calculations in [3.5], which becomes

$$\sum_{i=0}^{\infty} b\rho^i = \frac{b}{r} \quad (3.7)$$

The only further modification required is to change `bestVal()` such that in the final stage, it returns the present discounted value of future stages under mass selection.

3.6.8 Choice costs

The results so far consider only the revenue from the optimization program and not the costs. Realistically, information and/or institutional costs will exist; the optimization program should only continue when the benefits exceed the cost of the information, the benefit being the gain *in excess of* the gain from the default choice. A slight change in the algorithm allows the cost to be evaluated: the best possible choice is still found, but its value is compared to the value of switching to the default choice method. If it does not exceed the default choice by the information cost, the switch is made to the default choice.

3.6.9 Changing the horizon

The largest computational cost is not in calculating the values of the respective states, but in preparing the bubblettes for this computation and locating bubbles not on the initial path. Once it is known that states beyond a given stage are not used, there is no reason to continue calculating these states. Similarly, if a breeding program has not switched to mass selection, a longer breeding program may be desirable. The logical control variable `smartShrinkGens` is added to handle this situation.

With `smartShrinkGens` set to true, if default choices are not chosen in the final choice stage, the time horizon is increased by one. The solution of the current iteration, augmented by default choice for the final stage, is taken as the center, and the next iteration is run with the same grain.

Conversely, if the switch to default choice occurs before the final stage, the stage in which the switch occurs becomes the final stage. However, the grain is reduced, as the available states are a subset of the states already considered.

This final model can be expressed as

$$\max_{T, \{f_t: 0 \leq t \leq T-1\}} \left[\sum_{t=0}^{T-1} \rho^{t+1} V(S_{t+1}(f_t|S_t)) + \rho^T W(S_T) \right] \quad (3.8)$$

where $W(S_T)$ is the present value at time T of making default choices forever starting from state S_t .

3.7 Higher dimensions

While this method effectively solves single-dimensional problems, its true value lies in the ability to partition and search subsets of multidimensional space afflicted by Bellman's Curse of Dimensionality. It is again assumed that changes in the state can again be divided into cumulative and transitory effects.

3.7.1 Change in bubble and bubblette structure

The initial structure, which assumed a small number of bubbles which could be moved to handle boundary solutions, is not practical in multiple dimensions: too many moves may be needed. Instead new bubbles will be created and "misses" handled.

While there is no reason in principal that all bubbles have the same size or grain, in practice, requiring them to do so will allow faster searches. This is best illustrated by considering how the function `bubbletteAt()` works, as illustrated in Figure 3.12. `bubbletteAt()` is the key to the multi-dimensional search, as it is the method by which bubbles are found after "unexpected" transitions.

In the single-dimensional solution, bubblettes were referred to by references relative to the centers of their bubbles, which was practical when a bubble for a stage would always transition to the bubble for the next stage. With multiple possible destinations in different bubbles, this information is not useful for searching, and absolute references are used. With four-byte integers, it is possible to specify more than a billion states in each dimension; this is sufficient for nine digits of precision in the genetic models.

An array `bubCenters` is used to store the centers of the bubbles; and for the starting solution in any iteration, the centers can be any valid bubblette. Subsequent bubblettes are forced onto a grid, and may (and, unless the initial bubble was centered on the grid, will) overlap the starting bubbles. While the starting bubbles could also be forced onto this grid, this would move starting solution away from the center of the bubble, increasing the chances of a miss.

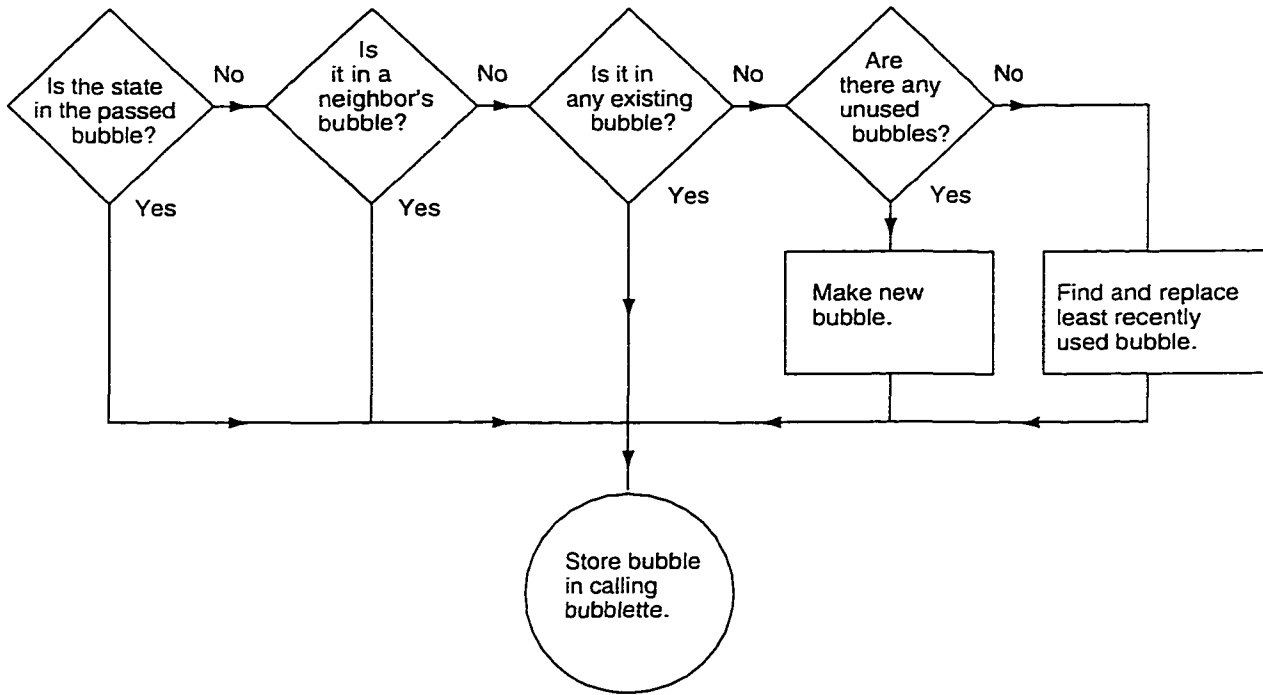


Figure 3.12 Searching for a bubble with `bubbletteAt()`. The function is passed the location in state-space that is needed, and an initial guess as to a bubble that might contain that state. The identity of the bubblette calling for the search is also known to the function.

While searching, `bubbletteAt()` is passed to the calling bubblette, the suspected next bubble, and the absolute address of the desired next bubble. `bubbletteAt()` first checks to see if the bubblette is in the suspected bubble, and quickly returns if this is the case. If not, all bubblettes *within the bubble that contains the calling bubblette* are checked for an indicator of the next bubble. These bubbles, if any, are checked for the bubblette. Failing this, the center that a bubble that contained such a bubblette *would* have if its bubble existed is checked against all bubbles until a match is found. If no match is found, a new bubble is created. In all cases, `bubbletteAt()` stores the bubble of the next bubblette in array `nextBubs` to speed subsequent searches.

Finally, it should be noted that the maximum number of bubbles allowed is set at compile time for performance reasons. It is possible that the algorithm will run out of bubbles. To guard against this possibility, each time `bubbletteAt()` is called, it increments a counter, and stores the value of the counter value with the bubble accessed. This allows the least recently used bubble to be identified and discarded if necessary, though the bubbles containing the starting solution will never be discarded.¹

3.7.2 Searching the multi-dimensional space

In large part, the multi-dimensional search is similar to the single-dimensional search. As such, only the differences will be discussed.

The first difference is in the pattern to search for states; there is no concept of "next" in multiple dimensions. Instead, a sequence is defined that includes all adjacent states. These are sequentially checked; and when a superior state is found, it becomes the new center of the search, which begins again.

Significant overlap of these search spaces is probable, and there is no practical way to store and quickly search all bubblettes already checked; this would require the impractical repeated and potentially expensive invocations of `bubbletteAt()`. Instead, `bestVal()` increments a counter each time it is called and stores this value in each bubblette it checks. This gives a high probability of determining whether or not it has already checked a bubblette, though not a guarantee, as a descendant invocation of the function may check the same bubblette. However, it is generally unlikely that more than one stage will look at the same bubblette.

Another difference arises in that with a single-dimension, it may be possible to find limits on the

¹It may be desirable to have "protected" sets of bubbles for each generation. For example, of 1,000 bubbles total, a decision may be made to set aside fifty bubbles for each of ten periods, such that only that period can overwrite those bubbles. This would guard against thrash conditions in which the an expansive search in a later period overwrites bubbles still in use by an early period. However, this would require search of both the common and individual areas, followed by a comparison, and has not been found necessary for the genetic problem.

extremal values of the state variable in a straightforward manner. With multiple dimensions, the limits on each state variable depend may depend upon the values of the others. As such, no attempt is made to handle such limits; instead, as a state is considered, an attempt is made to solve for the choice variables that yield that state. If no solution exists, the state is considered impossible.

3.7.3 Separable substates of the states

In some problems, it may be possible to separate the state variables for the next generation into substates that are fully or partially separable in the mathematical sense. For example, in the genetic problem of Chapter 6, the choice spaces for the sires and dams are separable and independent of each other. In such cases, thoughtful caching of derived values may yield immense rewards in reduced computational time.

Consider a two-part problem, in X and Y , which correspond to \bar{A}_{t+1}^s and \bar{A}_{t+1}^d in [6.5]. The choices made for X_t and Y_t produce substates $x(X_t)$ and $y(Y_t)$ that combine as

$$z_t(X_t, Y_t) = z(x(X_t), y(Y_t)) \quad (3.9)$$

That is, the specific values of X_t and Y_t do not interact. Suppose further that the functions $x()$ and $y()$ are reasonably expensive to compute and dependent upon the current state.

To make a search of all adjacent states would require that *each* possible value of X and Y occur *at least* three times: once for each state of the other. Furthermore, if movement is made in the X direction, the same value of Y will again be needed multiple times. It seems reasonable, then, to store these in some fashion.

While other methods are possible, the approach that will be taken in such situations is to have separate caches for each substate within each instance of `bestVal()`. The cache will be centered at the starting point of the search, and begins filled with an initial value indicating that it has not been calculated. When a value of $x(X_t)$ is needed, a check is first made that X_t is within the bounds of the cache, dimension by dimension. If not, the Fortran intrinsic function `EOSHIFT()` is called to cheaply move the cache by a set amount in the direction needed.

The benefits of the cache will depend upon the application. In the genetic problem, the payoff is significant, as calculating the polygenic gains is the second most expensive portion of the algorithm; bubble lookup is the most expensive.

CHAPTER 4 THE GENETIC PROBLEM IN ONE DIMENSION

4.1 Formulation

To establish a framework, a simplified version of the genetic model will be used, along with some strong assumptions necessary for the model's development. Particularly, an infinitely large breeding herd with an infinite number of genes is assumed, yielding a normal distribution of the breeding values of the animals. It is further assumed that genetic progress does not cause deviations from normality. The variance of the polygenic distribution will be assumed to remain constant for the current study, which is justified by Bulmer's findings that in response to selection, variance drops but asymptotically approaches a fixed value [Bulmer 71].

For the case of optimization of genetic improvement with a QTL, or identifiable Quantitative Trait Locus, the general model can be further specified. The time periods are the generations of breeding, and the state variables are the frequency of the major genes (including covariance), mean and variance of the polygenic distribution, the covariance between the polygenic effect and major genes, and the breeding value (which is actually a function of the others). While the major genes and the polygenic effect interact only additively in this example, any linear function would be permissible.

4.1.1 Simple case: one locus

In each generation, there will be the same number of "kinds" of creature, as determined by the combinations of alleles present at a single major gene. With a single locus having only two possible values (b and B), there are three types of creatures—namely those with 0, 1, or 2 of the superior gene (bb , bB , and BB). Generally, the number of types is the product of the number of permutations for each locus.

The choice set for each generation is the fraction of each kind of creature to be bred to produce the following generation. Let m represent the type of animal (0, 1, or 2), f_{mt} represent the fraction of type m that is chosen to breed at time t , and p_{mt} the fraction of all animals at time t that are of type m : the sum of the products of these fractions with the corresponding frequencies must equal Q , the fraction of

the entire population needed to produce the next generation:

$$\sum_m p_{mt} f_{mt} = Q \quad (4.1)$$

Note that these choices are not fully independent; if there are n types, the first $n - 1$ choices also determine the final choice. Additionally, while the fractions are necessarily in the $[0, 1]$ range, not all choices are necessarily possible. For example, if .2 of the population is needed to breed the following generation, and type m has a frequency of .4, f_{mt} must be chosen from $[0, .5]$.

Given the multivariate distribution of the major and polygenes and the frequencies of the kinds, the average polygenic breeding value of the herd may be calculated. Alternatively, and more easily, it can be calculated from its value in the prior generation. For example, with a single major gene, and ignoring gametic phase disequilibrium, the result is

$$\bar{A}_{t+1} = \bar{A}_t + \frac{h^2 \sigma_t}{Q} \sum_m q_{mt} z_{mt} \quad (4.2)$$

where z_{mt} is the height of the standard normal distribution at the truncation point [Dekkers 98], and q_{mt} is a function of p_{mt} specifying the portion of the herd that is of type m . Similarly, the average total genetic value of the herd, \bar{G}_t can be calculated from \bar{A}_t and the gene frequencies at time t as

$$\bar{G}_t = a(2p_t - 1) + \bar{A}_t \quad (2.10)$$

for additive genes, to which attention will be restricted until Chapter 7.

Finally, the objective function will typically be one of two forms: either a function, perhaps equality, of \bar{G}_T , the value after the final breeding, or a sum of a discounted profit function of the values of the state variables in each generation. The first is appropriate when the objective is to maximize genetic progress in a given number of generations. The latter calculates the economic value of a breeding program. Note that it is not necessarily the breeding value alone which is used but more generally a function of the breeding value. An example would be a price premium for animals which exceed a certain quality. In this case, the breeding value might represent the amount of meat produced, which could be sold at different prices depending upon how lean it is. Another example would be a fixed price premium for the fixation of a gene in the population, which would introduce a discontinuity into the objective function.

4.1.2 Simplified finite number of generations

In this simplified version, only the mean breeding value of the herd after the final breeding generation is considered. This is a test of the maximum rate of progress over a finite period, but it is rarelt

economically reasonable: it neglects both the sale of the animals during most of the program and the residual value of the operation at the end of the finite period of time. Further, the present value of the early generations should count for *more* than the final generation, rather than nothing other than their contribution to the final generation.

Nonetheless, this model is useful in developing the numerical methods, and ferrets out potential problems in the methods which follow. It also shows the maximum rate at which genetic progress *could* be made over a fixed number of generations. This can provide useful insight into problem formulation and solution reasonableness. Additionally, this is the problem for which a solution has been demonstrated, and is therefore useful as a check on the accuracy of the methods [Dekkers 98]. However, this is almost surely not the economic problem.

4.1.3 Discounted finite generations

The economic objective function is the net present value of all future profits, discounted for all periods considered in the model. It should be noted that within this research, the discounted finite generations problem is not in itself of interest; it is developed as a tool for the infinite horizon problem.

The simplest form is to discount the revenues of each generation, assuming that costs are fixed and that revenues are a linear function of average breeding value, e.g., that the breeding value represents items such as milk produced. In this case, the problem is to maximize total profit

$$\pi = \sum_{t=0}^T \rho^t \bar{G}_t \quad (4.3)$$

where ρ is the discount factor rate, and is equal to $1 - r$, where r is the interest rate. Note that the first generation can be left out of the summation, as profits during that generation are predetermined by the initial state. However, as more complicated cases may have variable costs from choices made, this generation will be left in for the sake of consistency.

This formulation, however, still only accounts for a very simple case of a single major gene affecting a single trait. Further, it does not take into account economic factors such as premiums received for lines that are fixed for the gene, profits, or the ability to terminate genetic testing as a choice variable. To account for such abilities, profitability should be considered. For example, consider a simple case in which a hog has both meat yield Y and leanness Z . At a fixed leanness, or quality, Z , revenues are presumably linear in Y . However, there is no *a priori* reason to believe that revenue is linear in leanness, though it would presumably be increasing within some range of interest, and then likely decreasing—no one wants to eat a chunk of fat with a bit of meat embedded, but without any fat, meat

is too dry to be enjoyable. Assuming that the desirability of leanness is unimodal, or that there is a single most desirable value, with desirability monotonically decreasing above and below this value, weak quasi-concavity should apply to revenues as a function of Z —it should be unimodal with a global maximum. Finally, it is possible that for a given set of genes for yield, that total meat produced may be different for different levels of leanness. Revenue then becomes a function of both leanness and yield, and the problem expands to be that of maximizing

$$\pi = \sum_{t=0}^T \rho^t \pi_t (Y_t, Z_t) \quad (4.4)$$

Note that this formulation includes the possibility that costs change with the size or leanness of the animal. Letting γ denote the vector of total genetic and breeding values, this becomes

$$\pi = \sum_{t=0}^T \rho^t \pi (\gamma_t) \quad (4.5)$$

Finally, some models will include a price premium or penalty based on the presence or absence of a gene. To remain general, let θ denote the vector of all gene frequencies and their covariances with each other and the polygenic effect, and the problem becomes

$$\pi = \sum_{t=0}^T \rho^t \pi (\gamma_t, \theta_t) \quad (4.6)$$

While these optimization problems express the profit function π as a function of the state variables, the determinism of the model means that the state variables themselves are functions of the choice variables, f_{mt} . Letting ϕ_t be the vector of fractions f_{mt} selected at time t , equation [4.6] becomes,

$$\pi = \sum_{t=0}^T (1 - t)^t \pi (\phi_t) \quad (4.7)$$

The final modification is to note that the cessation of testing for one or more genes may be included in the model. That is, it is entirely possible that a point may be reached at which the value of the information from continued testing is less than the test cost. Consideration will be limited to cases in which testing occurs in every generation until its value is less than its cost, after which testing is terminated. As such, the testing choice variable for a gene takes a whole number value. Letting τ_g indicate the the first generation without testing for gene g , and τ itself be the entire vector of stopping times for all major genes being tested, 4.7 becomes

$$\pi = \sum_{t=0}^T (1 - t)^t \pi (\phi_t, \tau) \quad (4.8)$$

It should be noted that selecting f_{mt} after testing stops is nonsensical; there is only one type (unknown major gene) after this point, from which all must be chosen.

4.1.4 Infinite horizon

This is the actual economic problem of interest. As an economic model, the herd should be assumed to continue forever—even though the farmer will eventually retire, the discounted value of the remaining infinite horizon reflects the value for which the herd can be sold. This problem is actually easier to solve analytically than the finite horizon—in the cases in which it *is* soluble. However, unless the problem can be analytically reduced to a single equation or function, it is not possible to solve for an infinite number of generations; a rule must be found for approximation of this horizon.

The problem is not as futile as it sounds. With the introduction of discounting, far-off generations have an increasingly diminished impact. Thus it can be expected that a convergence theorem can be written to the effect that for any desired ϵ , and for any generation s , that a total number of generations $T_{\epsilon,s}$ can be chosen sufficiently large that

$$|^{T_{\epsilon,s}} f_{mt}^* - f_{mt}^*| < \epsilon, \forall t \leq s \quad (4.9)$$

where f_{mt}^* is the optimal choice with an infinite horizon, and $^{T_{\epsilon,s}} f_{mt}^*$ is the optimal choice for a finite horizon of $T_{\epsilon,s}$ generations. On other words, a finite horizon can be chosen such that the error in the fractions selected is arbitrarily small.

Furthermore, the introduction of a generation in which to cease testing as a choice variable simplifies, rather than complicates, the problem. Once testing ends, selection is by mass selection, in which the animals to breed are selected solely on the observable value of the trait. The genetic progress under mass selection is well known[Falconer], and thus once the gene is fixed (or even not fixed, but testing ended), there is nothing new to the problem. A “canned” function can be written for the value of the entire future after the cutoff generation, as discussed above. This changes the problem from “find choices for all generations forever, ” with an infinite number of choice variables, to “choose a finite number of generations, and choices for those generations.”

Even without such a cutoff, the problem remains tractable. When the homozygote with both copies of the favorable gene is superior to the heterozygote, the gene frequencies rapidly approach one. In the case of overdominance, where the heterozygote is superior to either homozygote, a fixed value less than one will be approached instead. That is, after a very small number of generations, the gene frequency becomes very close to its optimal value, and remains so permanently. In the absence of overdominance, essentially all of the herd will have become homozygotes with the gene, and choosing the fraction for that type, the only one possible, becomes mass selection. Accordingly, an infinite horizon may be simulated by considering “enough” generations.

4.2 Optimal control methods for soluble cases

There are categories of cases which may be partially solved analytically, such as the case considered in [Dekkers 98]. However, even in this case, analytic methods fail to yield a complete solution, but instead take the problem to a point at which iterative methods can solve for the truncation points. This is as far as such a method can get.

However, the Dekkers solution is for a finite case. The actual economic problem is for the infinite horizon, in which the business continues indefinitely. The problem can easily be reformulated to include discounting and an infinite horizon. Consider again the Dekkers model described in Chapter 2. The genetic value G_t was defined in [2.10] as the sum of the effects of the major gene and the polygenic value

$$\bar{G}_t = a(2p_t - 1) + \bar{A}_t \quad (4.10)$$

where p_t is the frequency of the major gene, or the portion of loci in the population that actually have the favorable version of this gene, and a is the value of *each copy* of the gene. \bar{A}_t is the mean of the polygenic value, which is assumed to be normally distributed with standard deviation σ . The available choice variables are $\{f_{mt} : m \in \{0, 1, 2\}, t \in \{0, 1, \dots, T-1\}\}$, the fraction of type m that will be bred in generation t to produce the next generation. These are homo morphic with the truncation points x_{mt} and density z_{mt} of the standard normal distribution at the truncation points; transformations and inverses exist for translation of each of the three to any of the others. Breeding is by *truncation*: all animals better than the truncation point are bred, and are randomly assigned to another breeding animal. Q is the fraction of the entire population that must be bred to produce another population of the same size. The initial values, of p_0 and \bar{A}_0 , are known.

Dekkers' problem was to maximize the total genetic progress by the final generation T :

$$\max_{f_{mt}} \{L | \bar{A}_0, p_0, Q\} \quad (4.11)$$

(maximizing L by choosing the various choice variables f_{mt} for all relevant values of m and t , while taking \bar{A}_0 , p_0 , and Q as predetermined) where

$$\begin{aligned} L &= \sum_{t=0}^{T-1} \{H_t - \lambda_t p_t - \gamma_t \bar{A}_t\} - \lambda_T p_T + \lambda_0 p_0 - \gamma_T \bar{A}_0 + a(2p_T - 1) + \bar{A}_T \quad (4.12) \\ H_t &= \frac{\lambda_{t+1}}{Q} \{f_{1t} p_t^2 + f_{2t} p_t (1 - p_t)\} \\ &\quad + \gamma_{t+1} \left\{ \bar{A}_t + \frac{\sigma}{Q} \left[p_t z_{1t} + 2p_t (1 - p_t) z_{2t} + (1 - p_t)^2 z_{3t} \right] \right\} \\ &\quad + \epsilon_t \left\{ Q - f_{1t} p_t^2 - 2f_{2t} p_t (1 - p_t) - f_{3t} (1 - p_t)^2 \right\} \quad (4.13) \end{aligned}$$

[Dekkers 98, eq 6-9]. The Lagrangian multipliers for each period, λ_t , γ_t , and ϵ_t are used to ensure that equations [4.17], [4.18], and [4.17] are met.

The extension to an infinite horizon with discounting is straightforward. Using a constant discount value r , note that the total genetic value in generation t is \bar{G}_t and that the present value at time $t = 0$ is

$$(1 - r)^t \bar{G}_t = (1 - r)^t a (2p_t - 1) + \bar{A}_t \quad (4.14)$$

and the objective function to maximize becomes

$$G = \sum_{t=0}^{\infty} (1 - r)^t [a (2p_t - 1) + \bar{A}_t] \quad (4.15)$$

subject to

$$Q = f_{1t} p_t^2 + f_{2t} 2p_t (1 - p_t) + f_{3t} (1 - p_t)^2 \quad (4.16)$$

$$p_{t+1} = \frac{1}{Q} \{f_{1t} p_t^2 + f_{2t} p_t (1 - p_t)\} \quad (4.17)$$

$$\bar{A}_{t+1} = \bar{A}_t + \frac{\sigma}{Q} \{p_t^2 z_{1t} + 2p_t (1 - p_t) z_{2t} + (1 - p_t)^2 z_{3t}\} \quad (4.18)$$

where (4.16) is the constraint keeping population size constant, and (4.17) and (4.18) describe the progression of p and \bar{A} given the choices made and the current state. This formulation has the same constraints as the formulation developed by Dekkers and van Arendok [Dekkers 98], save only that Lagrange multipliers are required for an infinite number of time periods rather than a fixed number, a difference that will be critical below.

However, it is useful to rewrite this exclusively in terms of x_{mt} . Letting ϕ and Φ represent the probability density function (PDF) and cumulative density function (CDF), respectively, of the standard normal distribution,

$$f_{mt} = 1 - \Phi(x_{mt}) \quad (4.19)$$

$$z_{mt} = \phi(x_{mt}) \quad (4.20)$$

Equations (4.16-4.18) become

$$Q = (1 - \Phi(x_{1t})) p_t^2 + (1 - \Phi(x_{2t})) 2p_t (1 - p_t) + (1 - \Phi(x_{3t})) (1 - p_t)^2 \quad (4.21)$$

$$p_{t+1} = \frac{1}{Q} \{(1 - \Phi(x_{1t})) p_t^2 + (1 - \Phi(x_{2t})) 2p_t (1 - p_t)\} \quad (4.22)$$

$$\bar{A}_{t+1} = \bar{A}_t + \frac{\sigma}{Q} \{p_t^2 \phi(x_{1t}) + 2p_t (1 - p_t) \phi(x_{2t}) + (1 - p_t)^2 \phi(x_{3t})\} \quad (4.23)$$

$$\forall 0 \leq t < \infty \quad (4.24)$$

The Hamiltonian may be rewritten as

$$\begin{aligned}
H_t = & \frac{\lambda_{t+1}}{Q} \{ (1 - \Phi(x_{1t})) p_t^2 + (1 - \Phi(x_{2t})) p_t (1 - p_t) \} \\
& + \gamma_{t+1} \left\{ \bar{A}_t + \frac{\sigma}{Q} \left[p_t^2 \phi(x_{1t}) + 2p_t (1 - p_t) \phi(x_{2t}) + (1 - p_t)^2 \phi(x_{3t}) \right] \right\} \\
& + \epsilon_t \left\{ Q - (1 - \Phi(x_{1t})) p_t^2 - 2(1 - \Phi(x_{2t})) p_t (1 - p_t) - (1 - \Phi(x_{3t})) (1 - p_t)^2 \right\} \quad (4.25)
\end{aligned}$$

and the function to be maximized may be written

$$L = G + \sum_{t=0}^{\infty} \{ H_t - \lambda_t p_t - \gamma_t \bar{A}_t \} + \lambda_0 p_0 + \gamma_0 \bar{A}_0 \quad (4.26)$$

which differs from Dekkers' in the lack of a final period, and inclusion of the discounted values of all generations in G .

The partial derivatives of L are taken with respect to the choice variables x_{mt} , the state variables p_t and \bar{A}_t , and the Lagrangian multipliers, all of which derivatives must be equal to zero.

Taking the first partials of L yields

$$\begin{aligned}
\nabla_{x_t} L &= \nabla_{x_t} H \\
&= 0 \\
&= \frac{\lambda_{t+1}}{Q} \begin{bmatrix} -p_t^2 \phi(x_{1t}) \\ -p_t (1 - p_t) \phi(x_{2t}) \\ 0 \end{bmatrix} \\
&\quad - \frac{\gamma_{t+1} \sigma}{Q} \begin{bmatrix} p_t^2 x_{1t} \phi(x_{1t}) \\ 2p_t (1 - p_t) x_{2t} \phi(x_{2t}) \\ (1 - p_t^2) x_{3t} \phi(x_{3t}) \end{bmatrix} + \epsilon_t \begin{bmatrix} p_t^2 \phi(x_{1t}) \\ 2p_t (1 - p_t) \phi(x_{2t}) \\ (1 - p_t^2) \phi(x_{3t}) \end{bmatrix} \quad (4.27)
\end{aligned}$$

For p_t ,

$$\begin{aligned}
\frac{\partial L}{\partial p_t} &= 2(1 - r)^t a - \lambda_t \\
&+ \frac{\lambda_{t+1}}{Q} \{ 2(1 - \Phi(x_{1t})) p_t + (1 - \Phi(x_{2t})) (1 - 2p_t) \} \\
&+ 2 \frac{\gamma_{t+1} \sigma}{Q} \{ p_t \phi(x_{1t}) + (1 - 2p_t) \phi(x_{2t}) - (1 - p_t) \phi(x_{3t}) \} \\
&+ 2\epsilon_t \{ -(1 - \Phi(x_{1t})) p_t - (1 - \Phi(x_{2t})) (1 - 2p_t) + (1 - \Phi(x_{3t})) (1 - p_t) \} \quad (4.28)
\end{aligned}$$

The partial with respect to \bar{A}_t yields information about γ_t , its shadow value

$$\frac{\partial L}{\partial \bar{A}_t} = (1 - r)^t + \gamma_{t+1} - \gamma_t \quad (4.29)$$

while λ and ϵ yield only the dynamic behavior of p_t and the breeding constraint:

$$L_{\lambda_{t+1}} = \frac{1}{Q} \{f_{1t}p_t^2 + f_{2t}p_t(1-p_t)\} - p_{t+1} \quad (4.30)$$

$$L_{\epsilon_t} = Q - f_{1t}p_t^2 - f_{2t}2p_t(1-p_t) - f_{3t}(1-p_t)^2 \quad (4.31)$$

$$L_{\gamma_{t+1}} = \bar{A}_t + \frac{\sigma}{Q} [p_t^2 \phi(x_{1t}) + 2p_t(1-p_t)\phi(x_{2t}) + (1-p_t)^2 \phi(x_{3t})] - \bar{A}_{t+1} \quad (4.32)$$

Unlike the Dekkers formulation, there is no final period; no derivatives are calculated for that special case.

As all of the preceding equations are for derivatives which must be equal to zero at the optimum, [4.29] means that

$$\gamma_{t+1} = \gamma_t - (1-r)^t \quad (4.33)$$

This result is drastically different than Dekkers', which found that $\gamma_t = 1 \forall t$. As γ_t is the shadow value for \bar{A}_t , and the initial value \bar{A}_0 is known, it follows that γ_t is a known value that can be directly calculated. As the growth in polygenic mean \bar{A}_t is additive, \bar{A}_t contributes its own value to the undiscounted value function in the current and each subsequent generation. This value can be discounted to time 0, yielding

$$\frac{\partial G}{\partial \bar{A}_t} = \frac{(1-r)^t}{r} \quad (4.34)$$

and

$$\gamma_t = \frac{(1-r)^t}{r} \quad (4.35)$$

This relation can also be found by starting with the limiting value at infinity of zero for γ_t , and writing γ_0 as an infinite sum of the later values.

From [4.27] and [4.35] come the equations

$$-\frac{\lambda_{t+1}}{Q} \begin{bmatrix} 1 \\ .5 \\ 0 \end{bmatrix} + \frac{(1-r)^t \sigma}{rQ} \begin{bmatrix} x_{1t} \\ x_{2t} \\ x_{3t} \end{bmatrix} = \epsilon_t \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (4.36)$$

which may be solved

$$x_{3t} - x_{2t} = \frac{r}{(1-r)^t} \frac{\lambda_{t+1}}{2\sigma} \quad (4.37)$$

$$x_{2t} - x_{1t} = \frac{r}{(1-r)^t} \frac{\lambda_{t+1}}{2\sigma} \quad (4.38)$$

Which gives the result that

$$x_{3t} - x_{2t} = x_{2t} - x_{1t} \quad (4.39)$$

or that the truncation points are equidistant, as Dekkers found in the finite model for additive major genes.

In the limiting case of very steep discounting, as r approaches one and ρ approaches zero, only the first generation which can be controlled should matter, as all later generations are discounted so steeply as to not matter. The numerators of the discount factors in equations [4.37] and [4.38] approach one, while the denominators are identically equal to one for the initial generation in which t is equal to zero, and the equations become the same as those for the final generation in Dekkers' paper, for which the future does not matter.

The relations of [4.36] can be used to remove the ϵ_t from [4.28], which becomes

$$\begin{aligned}
\lambda_t &= 2(1-r)^t a \\
&+ \frac{\lambda_{t+1}}{Q} \{2(1-\Phi(x_{1t}))p_t + (1-\Phi(x_{2t}))(1-2p_t)\} \\
&+ 2\frac{(1-r)^t \sigma}{rQ} \{p_t \phi(x_{1t}) + (1-2p_t)\phi(x_{2t}) - (1-p_t)\phi(x_{3t})\} \\
&- \frac{\lambda_{t+1}}{Q} \{2(1-\Phi(x_{1t}))p_t + (1-\Phi(x_{2t}))(1-2p_t)\} \\
&- 2\frac{(1-r)^t \sigma}{rQ} \\
&\quad \{(1-\Phi(x_{1t}))x_{1t}p_t + (1-\Phi(x_{2t}))x_{2t}(1-2p_t) - (1-\Phi(x_{3t}))x_{3t}(1-p_t)\} \\
&= 2(1-r)^t a + 2\frac{(1-r)^t \sigma}{rQ} \{[\phi(x_{1t}) - (1-\Phi(x_{1t}))x_{1t}]p_t \\
&\quad + [\phi(x_{2t}) - (1-\Phi(x_{2t}))x_{2t}](1-2p_t) - [\phi(x_{3t}) - (1-\Phi(x_{3t}))x_{3t}](1-p_t)\} \quad (4.40)
\end{aligned}$$

Which is an expression, for x_{mt} and λ_t .

Unfortunately, this expression is not amenable to analytic or numeric solution. For any given time period, equations [4.16], [4.37], [4.38], and [4.40] produce four equations in five unknowns for any given time period: the three x_{mt} , and the multipliers λ_t and λ_{t+1} . If λ_t were known for any finite period, the problem would be soluble for all periods. The limiting value of λ_t as t approaches infinity is zero, which is nothing more than the standard result that there is very little value of something that far away. It is therefore not possible to do an infinite summation such was done to produce [4.35], as the remaining terms for each period are combinations of the CDF and PDF of the standard normal distribution preventing a closed form summation.

The only remaining possibility would be to find a value for λ_0 . While the expression

$$\lambda_0 = \frac{d}{dp_0} G \quad (4.41)$$

can be written, this is of little value, as calculating this value requires knowledge of the solution. As

such, there appears to be no solution available through Lagrange multipliers for the pure infinite horizon problem.

4.3 Infinite horizon with testing costs

While the simple problem of the infinite horizon cannot be treated analytically with the methods used for the finite horizon, because of the inability to find an initial Lagrangian multiplier for some finite time period, the introduction of a cost for genetic testing makes the problem tractable.

Animals do not come with labels on their foreheads indicating their genetic makeup; if so, a sample of the the polygenic distribution would be known as well. Instead, there is some finite cost for the genetic testing of an animal. Moreover, testing should not occur if the gains are outweighed by the costs.

Conveniently, it is possible to put a ceiling on the value of testing. The greatest possible gain from the major gene is for it to become fully present in the population: That is, for p_t to change from its present value to 1 (Note that this does not hold over over-dominant genes, which are not considered herein). Consider first the extreme case of taking this entire gain in a single generation. If no selection at all were to occur after this generation, the gene would keep its current frequency. Thus the change for the next generation is $1 - p_t$, which has a value of $2a(1 - p_t)$ in the subsequent generation. Further, compared to the case of no selection, it has that value in all future generations. Thus, the present value of the change is

$$\begin{aligned} PV &= \sum_{i=1}^{\infty} (1-r)^i 2a(1-p_t) \\ &= 2a(1-p_t) \frac{1-r}{r} \end{aligned} \quad (4.42)$$

Thus, in no case is it worth testing the animals if the cost c of testing is greater than this value, or for an initial critical value p^c of gene frequency

$$p^c = 1 - \frac{cr}{2a(1-r)} \quad (4.43)$$

Note that this is a conservative limit and only a starting point for the critical value. It would be possible to use this as a starting value for a value \hat{p} that could be updated—if testing is unprofitable for any value p_t , it would also be unprofitable for any larger value. This course is not pursued herein as it is not practical for higher dimensions, as separate critical values would be needed for each value of the other state variables.

However, even if selection on the major gene is not profitable, mass selection would still presumably be used, allowing a tighter limit to be drawn. Mass selection will continue to cause the frequency of the major gene to change; animals with this gene will be selected at lower polygenic values, and the gene will become relatively more common in the population. Without testing, the observed or phenotypic truncation point will be the same for all groups, with the result that the polygenic truncation points are separated by exactly a , or that

$$x_{1t} + \frac{h^2 a}{\sigma} = x_{2t} = x_{3t} - \frac{h^2 a}{\sigma} \quad (4.44)$$

The frequency in the next generation as a function of the current frequency can then be calculated. First, the overall truncation point is

$$x_t = \Phi^{-1}(1 - Q) \quad (4.45)$$

That is, all creatures in the upper fraction Q of the population are kept. Given that the program of breeding for mass selection is well known, it is possible to write a value function for the current value of the future gains from mass selection,

$$PV_m(p_t) = \sum_{i=1}^{\infty} (1 - r)^i [2a(p_{t+i}^m - p_t) + (\bar{A}_{t+i}^m - \bar{A}_t)] \quad (4.46)$$

The amount of work to be invested in finding a better bound will depend upon how much computation the bound saves; a loose bound does no harm, but merely requires additional computation.

The bound has only one real purpose: if the maximum gain from all future testing exceeds this generation's cost of testing, there is no reason to test, and mass selection will be used forevermore.

4.4 The breeding problem and dynamic programming

The naive approach to the breeding problem would be to look at the solution spaces as $\{f_{mt}\}$. However, this is far too complicated a space to use as a state space for dynamic programming: at even a resolution of .01, there are 100 possible values for each choice, or 10^4 per generation with only one choice gene. Then for T generations, there are 10^{4T} possible states to consider.

However, for a finite horizon problem, or an infinite horizon problem with a known \bar{p} bound such as [4.43], p_t alone may be used as the state. p may be divided into as many states as desired, and the optimal choice for each value of p_t for each generation t can be calculated.

Consider again the nature of the general problem: once a change has been made to \bar{A}_t , the change is permanent. The optimal choice depends only upon the present value of p_t and the number of

generations remaining. The optimal behavior for $p_t > \hat{p}$ is known, namely to switch to mass selection. This is entered into the action space as a beginning. Letting

$$\Delta = \frac{1}{S} \quad (4.47)$$

where Δ is the spacing between potential values of p_t for a number of states S , the first action considered is for $p_t = \hat{p} - \Delta$. There are only two choice variables to consider, f_{1t} and f_{2t} . With only one state variable, the choice of f_t is not unique. To resolve this, f_t is selected so as to maximize the change in polygenic breeding value.

For a single state variable p_t , this appears to be an efficient search: rather than checking all possible values of p_t , only a portion are checked. While this is convenient for the single state variable, it maybe critical as the number of state variables increases.

4.5 An initial algorithm

Consider first the simple problem of one gene with a fixed polygenic variance. For any given p_t and \bar{A}_t , there is an associated optimal value $G(p_t)$, reflecting the greatest value of the objective function, whether present discounted value or the total breeding value of the final generation, that can be achieved starting with p_t . First, using [3.7] note that for an infinite horizon, the starting value \bar{A}_t affects the present value by only a known offset

$$G(p_t, \bar{A}_t) = G(p_t, 0) + \frac{\bar{A}_t}{r} \quad (4.48)$$

Then break G into three pieces: the value of generation t , which choices in t cannot affect (Equation[2.10]); the discounted value of the subsequent generation $t + 1$; and the discounted value of all generations after $t + 1$:

$$\begin{aligned} G(p_t, \bar{A}_t) &= \bar{G}(p_t, \bar{A}_t) + (1-r)\bar{G}(p_{t+1}, \bar{A}_{t+1}) + \sum_{s=t+2}^{\infty} (1-r)^s \bar{G}(p_t, \bar{A}_t) \\ &= \bar{A}_t + a(2p_t - 1) \\ &\quad + (1-r) \left[\frac{\sigma}{Qr} \left\{ 2p_{t+1}^2 z_{1t+1} + 2p_{t+1}(1-p_{t+1})z_{2t+2} + (1-p_{t+1})^2 z_{3t+1} \right\} + \bar{A}_t \right] \\ &\quad + (1-r)a \frac{1}{Q} \left\{ 2f_{1t}p_t^2 + f_{2t}p_t(1-2p_t) \right\} \\ &\quad + \sum_{s=t+2}^{\infty} (1-r)^s [a(2p_t - 1) + \bar{A}_s] \end{aligned} \quad (4.49)$$

Note that part of the value of \bar{A}_s in the summation can be regrouped into the first term, as can the appearance of \bar{A}_t in the period $t + 1$. However, another approach will be more fruitful, namely redefining

the objective function. Consider

$$\begin{aligned}
 F(p_t) \equiv & \frac{\sigma}{Qr} \left\{ 2p_t^2 z_{1t} + 2p_t(1-p_t) z_{2t} + (1-p_t)^2 z_{3t} \right\} \\
 & + \frac{a}{Q} \left\{ 2f_{1t} p_t^2 + f_{2t} p_t(1-2p_t) \right\} \\
 & + (1-r) F(p_{t+1})
 \end{aligned} \tag{4.50}$$

As

$$G(p_t) = \frac{\bar{A}_t}{r} + a(2p_t - 1) + (1-r) F(p_{t+1}(p_t)) \tag{4.51}$$

maximizing F is equivalent to maximizing G .

The problem is now in a state to which dynamic programming can be applied. Using S possible states for p_{t+1} , let $f_t(p_t, p_{t+1})$ be the value of f_t that maximizes $\Delta \bar{A}_{t+1}$ for the indicated values of p , where

$$\Delta \bar{A}_t = \bar{A}_{t+1} - \bar{A}_t \tag{4.52}$$

The optimal values for $f_t(p_t)$ are simply the allowed states for which G is greatest.

The availability of a value for $F(p_s)$ for all s larger than t is taken as a given; it will be called recursively if necessary. This reduces the problem to finding the best set of f_t for the current generation. An initial trial solution is considered, from whatever source, yielding a tentative value for of the next frequency given the current frequency, p_{t+1}^t . $F(p_{t+1}^t)$ is treated as known, and the remainder of $F(p_t)$ is optimized over $\{f_t : p_{t+1}(p_t, f_t) = p_{t+1}^t\}$, for which standard optimization methods suffice. This is compared to the maximum over $\{f_t : p_{t+1}(p_t, f_t) = p_{t+1}^t + \Delta\}$, and the better value is stored.

The optimal value and behavior with granularity Δ can now be found by simply calling this optimal value function for $F(p_0)$, which calls itself recursively to find any other needed values for other values of p_t . Many possible states will likely be passed over, saving computational time. These solutions can then be used as starting values for another run with a finer grain, until the desired level of resolution is found. The algorithm has previously been described in Figure 3.10 at page 30.

As a side effect of this method, the stiff matrix problem with genetic algorithms and Newton methods is avoided entirely: there is never an attempt to directly determine the present value of the effect of a change in a current variable on a state several generations away; all calculations find the optimal value by looking ahead only one generation, rather than working with values for distant generations.

4.5.1 Simple dynamic programming algorithm

For an initial algorithm to demonstrate the concepts presented herein, consider again the simple model of [4.11]. As this model has a known solution, it will be the first to be solved. However, in

designing structures and methods, more attention will be paid to the usefulness of the model in solving the general case than the problem at hand.

Using equations [4.16], [4.17], and [4.18],

$$Q = f_{1t}p_t^2 + f_{2t}2p_t(1-p_t) + f_{3t}(1-p_t^2) \quad (4.16)$$

$$p_{t+1} = \frac{1}{Q} \{f_{1t}p_t^2 + f_{2t}p_t(1-p_t)\} \quad (4.17)$$

$$\bar{A}_{t+1} = \bar{A}_t + \frac{\sigma}{Q} \{p_t^2 z_{1t} + 2p_t(1-p_t) z_{2t} + (1-p_t)^2 z_{3t}\} \quad (4.18)$$

[4.17] can be manipulated into

$$f_{2t}(f_{1t}|p_{t+1}) = \frac{Qp_{t+1} - f_{1t}p_t^2}{p_t(1-p_t)} \quad (4.53)$$

expressing f_{2t} as a function of f_{1t} taking p_{t+1} as given. [4.16] becomes

$$\begin{aligned} f_{3t}(f_{1t}|p_{t+1}) &= \frac{Q - f_{1t}p_t^2 - f_{2t}(f_{1t}|p_{t+1})2p_t(1-p_t)}{(1-p_t)^2} \\ &= \frac{Q - f_{1t}p_t^2 - \frac{Qp_{t+1} - f_{1t}p_t^2}{p_t(1-p_t)}2p_t(1-p_t)}{(1-p_t)^2} \\ &= \frac{Q + f_{1t}p_t^2 - 2Qp_{t+1}}{(1-p_t)^2} \end{aligned} \quad (4.54)$$

and [4.18] becomes

$$\bar{A}_{t+1} = \bar{A}_t + \frac{\sigma}{Q} \left\{ \begin{array}{l} p_t^2 \phi(\Phi^{-1}(1-f_{1t})) \\ + 2p_t(1-p_t) \phi\left(\Phi^{-1}\left(1 - \frac{Qp_{t+1} - f_{1t}p_t^2}{p_t(1-p_t)}\right)\right) \\ + (1-p_t)^2 \phi\left(\Phi^{-1}\left(1 - \frac{Q + f_{1t}p_t^2 - 2Qp_{t+1}}{(1-p_t)^2}\right)\right) \end{array} \right\} \quad (4.55)$$

This allows the translation of the problem into a one-dimensional problem for purposes of dynamic programming. The sole state variable is p_{t+1} , and [4.55] is optimized with respect to f_{1t} for the chosen p_{t+1} . To make the calculation useful for dynamic programming, it is actually the change, the maximal value of

$$\Delta \bar{A}_t(f_{1t}) = \bar{A}_{t+1}(f_{1t}) - \bar{A}_t \quad (4.56)$$

that is calculated, rather than \bar{A}_{t+1} itself, so that the calculated value of selecting p_{t+1} is useful regardless of the prior value of \bar{A}_t , which will vary with the choices made in periods prior to t .

Given that the available numeric libraries minimize rather than maximize, a new function y is defined as

$$\begin{aligned}
y &= -\frac{Q}{p} (\bar{A}_{t+1} - \bar{A}_t) \\
&= -p_t^2 z_{1t} - 2p_t (1 - p_t) z_{2t} - (1 - p_t)^2 z_{3t} \\
&= -p_t^2 z_{1t} - 2p_t (1 - p_t) z \left(\frac{Q p_{t+1} - f_{1t} p_t^2}{p_t (1 - p_t)} \right) \\
&\quad - (1 - p_t)^2 z \left(\frac{Q + f_{1t} p_t^2 - 2Q p_{t+1}}{(1 - p_t)^2} \right)
\end{aligned} \tag{4.57}$$

with the derivative

$$\begin{aligned}
\frac{dy}{df_{1t}} &= -p_t^2 x(f_{1t}) - 2p_t (1 - p_t) x(f_{2t}(f_{1t})) \frac{df_{2t}}{df_{1t}} \\
&\quad - (1 - p_t)^2 x(f_{3t}(f_{1t})) \frac{df_{3t}}{df_{1t}} \\
&= -p_t^2 \Phi^{-1}(1 - f_{1t}) - 2p_t (1 - p_t) \Phi^{-1}(1 - f_{2t}(f_{1t})) \frac{-p_t^2}{p_t (1 - p_t)} \\
&\quad - (1 - p_t)^2 \Phi^{-1}(1 - f_{3t}(f_{1t})) \frac{p_t^2}{(1 - p_t)^2} \\
&= -p_t^2 \Phi^{-1}(1 - f_{1t}) + 2p_t^2 \Phi^{-1}(1 - f_{2t}) \\
&\quad - p_t^2 \Phi^{-1}(1 - f_{3t})
\end{aligned} \tag{4.58}$$

When y is minimized, the increase in \bar{A}_{t+1} is maximal for a given p_t and p_{t+1} .

Note that while the derivative of y has been calculated, this may not be possible for some models. However, the optimization routines in standard libraries are both faster and more efficient when this derivative is available.

Also note that the conversion of the problem to use discrete state space yields a different starting point than the actual mass selection solution. While the frequencies are as near to those reached by mass selection as permitted by the starting discretization, the f_t are actually different, as they are selected to maximize \bar{A}_{t+1} in the subsequent generation. As such, the starting point is actually a superior solution to the mass selection solution.

4.5.2 Solving the model

At this stage, the goal is a clean and easily understood routine, and no attempts have been made at optimization of the algorithm. The one-dimensional problem is a necessary first step towards the multidimensional algorithm of interest, and follows the flowchart of Figure 3.10 except as noted.

To provide the initial configuration for the algorithm described above, the mass selection solution is taken as the starting value. An array of bubbles is created with indices from zero to the total number of

generations considered, `totGenerations`. Each of these bubbles is centered around the corresponding mass selection gene frequency. Bubble 0 has a single bubblette, as the starting frequency is a parameter of the problem. Each of the other bubbles receive bubblettes to reach from .05 beneath the mass selection value to as close as possible to 1.0 without exceeding it, and a grain of .05 is used for all generations. This is far larger than is needed, but by extending beyond all conceivable values, issues related to stepping past boundaries are avoided for the moment.

The function `bestVal()` is defined, which returns the best possible increase in the value of the objective function that can be reached from a specified bubble. This is the change in value from that particular choice of frequency at that generation, *including the value from all subsequent choices*.

The function works by sequentially considering a subset of the possible choices for the *next* period's major gene frequency. However, before it does so, it checks to see if this bubblette has already been considered. If so, it returns the previously calculated value. Failing this, an initial guess, which will for the moment and for illustrative purposes be assumed to be 0, as to the offset from the center of the next bubble, or how many allowed states away from the center of the bubble, is made. The corresponding bubblette in the next generation is queried for its value, to which the gain $\Delta \bar{A}_{t+1}$ resulting from this tentative choice is added, and the result placed in `testVal`. `testVal` is also copied to the local variable `thisBest`, which stores the best value of the best step found to date.

The routine then evaluates an offset one greater than the first considered and calculates its value in the same way. If the result is better than that stored in `thisBest`, `thisBest` is updated, and the process repeated for the next larger offset.

If the result of the first step is inferior to `thisBest`, the direction is switched; the offset one less than the starting value is considered.

The process is repeated until the next value considered declines. At this time, the prior step is recognized as the best, the step stored in `nextp`, and the value returned.

It is important to note at this time that the bubblette with the highest value will not necessarily be chosen. In fact, it seems that this happens only on rare occasions. As described in §3.4.3 it is not only the value of the bubblette itself that matters, but also the value *added* in stepping to that bubblette. As explained earlier, adding .5 while reaching a bubblette with value .3 has greater value than adding .2 while reaching a bubblette with value .4. Additionally, though the value of the bubblette is fixed regardless of the step taken to reach it, the value added reaching it will vary depending upon the prior state. Thus different bubblettes will be chosen from different prior states. In non-trivial problems, it is expected that there will be a tradeoff between the stepping gain and the value of the bubblette; if this

were not the case, both could be maximized.

`bestVal()` does have a “special” case for the final period. The last choice is made in the penultimate generation. `bestVal()` is called again, but recognizes that it is called for the final generations, and simply returns $(2p - 1)a$, the contribution of the major gene in the final period.

The function operates recursively. As such, querying the sole bubblette in generation 0 for its value results in calculation of needed values for all subsequent generations. When a solution is reached, the bubbles are “collapsed.” New bubbles are created centered about the path chosen, and with the grain reduced. Additionally, hints are stored regarding the “expected” path from bubblette to bubblette. For the center bubblettes, this is simply the next centered bubblette. For other bubblettes *in the region calculated in the prior iteration*, the hint is the bubblette at the same point in space which was previously reached. Due to the collapsing, there will be somewhat arbitrary choices made as to which bubblettes of the prior iteration correspond to which in the new. However, rather than spend effort and computation on the matter, this is simply left to the default rounding performed by Fortran: at most, it will be off by a single state in any dimension, and adjacent states will be checked in any event.

A minimum of two bubblettes above and below the center will be created within each bubble. However, a check is made to determine how many states were actually checked in that direction for that bubble, and if larger than two, this quantity is used instead. Furthermore, a check is made to insure that no boundary states were chosen in the final solution—this could indicate that a better state existed after the boundary. In this case, the boundary is doubled, and the iteration repeated with the same grain. This is different than the algorithm used for multiple state variables as described in Figure 3.10, for which additional bubbles are created when a boundary is reached.

The process is repeated until the grain of the choice variables are satisfactorily small, with the final iteration yielding the reported solution.

Using the equations from [Dekkers 98], it is possible to use standard iterative methods to find the optimal values. However, these equations do not solve the problem, but rather reduce it to a point at which iteration may be used to find the solution. The algorithm reaches the same result as Dekkers’ method. Both outputs are reproduced in Appendix C.

4.5.3 Mechanics of conversion to frequency space

Building on the foundation laid in §4.5.2, precise rules can be designed to finish the conversion of the model from the choice variables f_t to a state space model. As discussed in §3.6.1, a fundamental change to the problem has been made while converting from choice space to frequency space, and the

reduction from a two-dimensional search space to a single-dimensional space. While the same answer will be reached as the optimal solution, the starting point is not the same as the mass selection solution.

With p_{t+1} chosen, f_t should be chosen to maximize $\Delta\bar{A}_{t+1}$. The optimization problem is

$$\begin{aligned} \max_{f_t} L &= \Delta\bar{A}_{t+1} \\ &+ \lambda [f_{1t}p_t^2 + f_{2t}p_t(1-p_t) - Qp_{t+1}] + \mu [f_{1t}p_t^2 + f_{2t}2p_t(1-p_t) + f_{3t}(1-p_t^2)] \\ &= \frac{\sigma}{Q} \{p_t^2 z_{1t} + 2p_t(1-p_t) z_{2t} + (1-p_t^2) z_{3t}\} \\ &+ \lambda [f_{1t}p_t^2 + f_{2t}p_t(1-p_t) - Qp_{t+1}] + \mu [f_{1t}p_t^2 + f_{2t}2p_t(1-p_t) + f_{3t}(1-p_t^2)] \end{aligned} \quad (4.59)$$

where the constraint associated with λ is the transition rule for p_{t+1} from equation [4.17], and that with μ is the constraint that a total fraction Q be selected from [4.17]. Solving this would require an iterative solution, due to the presence of inverses of the normal cumulative distribution function. As such, it is simpler to use a routine from a standard library. Still, such routines require boundaries for the search.

f_{1t} will be used as the choice variable in the optimization, with f_{2t} determined by this choice. In any "sane" choice,¹

$$f_{1t} > f_{2t} \quad (4.60)$$

In the general case, a lower bound for f_{1t} is found by setting [4.60] to equality. Using the superscript "0" to denote an initial floor for the value, the initial limit simplifies to

$$f_{1tmin}^0 = Q \frac{p_{t+1}}{p_t} \quad (4.61)$$

For sufficiently large p_t , setting f_{1t} and f_{2t} equal causes selection of more than Q . In this case, f_{1t} must be increased and f_{2t} decreased. In this case, the new floor will still allocate nothing, or as close to nothing as allocated by the algorithm, to f_{3t} . As such, using the superscript "1" to denote the alternate bound, [4.16] reduces to

$$f_{1tmin}^1 p_t^2 + f_{2tmin}^1 2p_t(1-p_t) = Q \quad (4.62)$$

which combined with [4.17] yields two equations in two unknowns for the value of f_{1t} that minimizes y as defined in [4.57]. As f_{1t} rises from this value, f_{2t} must diminish, and f_{3t} will rise. Solving the equations,

$$\begin{bmatrix} p_t^2 & 2p_t(1-p_t) \\ p_t^2 & p_t(1-p_t) \end{bmatrix} \begin{bmatrix} f_{1t} \\ f_{2t} \end{bmatrix} = \begin{bmatrix} Q \\ Qp_{t+1} \end{bmatrix}$$

¹This is not true for over-dominant genes, in which case this constraint is left out. However, the constraint only speeds calculation by limiting the domain and is not actually necessary.

$$\begin{aligned}
\begin{bmatrix} p_t & 2(1-p_t) \\ p_t & (1-p_t) \end{bmatrix} \begin{bmatrix} f_{1t} \\ f_{2t} \end{bmatrix} &= \frac{Q}{p_t} \begin{bmatrix} 1 \\ p_{t+1} \end{bmatrix} \\
\begin{bmatrix} f_{1t} \\ f_{2t} \end{bmatrix} &= \frac{Q}{p_t} \frac{\begin{bmatrix} (1-p_t) & -2(1-p_t) \\ -p_t & p_t \end{bmatrix}}{p_t(1-p_t) - p_t^2(1-p_t)} \begin{bmatrix} 1 \\ p_{t+1} \end{bmatrix} \\
&= \frac{Q}{p_t} \frac{\begin{bmatrix} (1-p_t) & -2(1-p_t) \\ -p_t & p_t \end{bmatrix}}{p_t(1-p_t)} \begin{bmatrix} 1 \\ p_{t+1} \end{bmatrix} \\
&= -\frac{Q}{p_t^2(1-p_t)} \begin{bmatrix} (1-p_t) - 2(1-p_t)p_{t+1} \\ p_t + p_t p_{t+1} \end{bmatrix} \\
&= -\frac{Q}{p_t^2(1-p_t)} \begin{bmatrix} (1-p_t)(1-2p_{t+1}) \\ p_t(1+p_{t+1}) \end{bmatrix} \\
&= \begin{bmatrix} \frac{Q(2p_{t+1}-1)}{p_t^2} \\ \frac{Q(1+p_{t+1})}{p_t(1-p_t)} \end{bmatrix} \tag{4.63}
\end{aligned}$$

of which only the first is used.

An upper bound must similarly be found. For

$$p_t < \sqrt{Q} \tag{4.64}$$

it is possible to use all of the homozygotes, and

$$f_{1tmax}^0 = 1 \tag{4.65}$$

For larger values of p_t ,

$$f_{1tmax}^1 = \frac{Q}{p_t^2} \tag{4.66}$$

selects entirely from type 1, and is used as the maximum. With these calculations done, the IMSL routine DUV MID() is used to solve the upper row of [4.63] for f_{1t} .

For this single state variable model, each generation has its own bubble; the bubble for any successor state chosen is known with certainty. For the initial pass, sufficient bubblettes are used such that the full range (0, 1) is available in each choice generation, save that a single bubblette is available for generation 0, as its gene frequency is predetermined as part of the problem.

An initial grain of .05 is set for p_t in all generations, and the bubblette of generation 0 is asked for its value.

When a bubblette is asked for its value, it firsts checks to see if it has been previously calculated. If so, it returns its value. If not, it checks itself for a hint from prior iterations, or runs of the search algorithm, choosing a hint of 0 if none is found. Having established the hint, a “sanity check” is made on the hint: the gene frequency will never be permitted to decrease from generation to generation; any such step is wrong.² As such, if the hint suggests reducing the next generation’s frequency below the present level, it is rejected, and increased to a “sane” level. Similarly, a sanity check is made to insure that the p_{t+1} considered is actually a possible transition.

This achieved, the value of the hint is calculated. The bubblette from the next generation corresponding to the hint is queried, and $\Delta \bar{A}_{t+1}$ is added to this value. This is stored as the best tentative value, and the next higher bubblette in the next generation is checked. If that bubblette is better, it becomes the new hint, and the search proceeds in that direction as long as progress is made. If inferior, the next lower bubblette is checked, and the search proceeds in that direction as long as it is successful. However, if the initial hint is less than 0, either from a sanity check or the initial hint, the search starts first in the negative direction and switches to positive if appropriate.

Once a bubblette selects a value in this manner, it stores the final hint, indicating the next state chosen, its value, and the fact of calculation. It then returns its value as the result of `bestVal()`. After the initial bubblette returns a value, the optimal path can be found by stepping from bubble to bubble, which is the method used by `candidateFromBubbles()` to form a reportable solution.

With a solution found, another iteration is made. If any of the steps chosen as optimal were on the boundary of a bubblette, it is not clear that a further step was not desirable. That boundary is doubled and the process repeated with the same grain. If no boundary states were used, the optimal solution becomes the base frequency of the next solution.

Initially, the boundaries were set by doubling one more than the highest state actually used in that direction (above or below the base), with a minimum of ten. Using one processor of a dual Pentium II/333 machine, this resulted in a solution time of 40.1 seconds (also using zero rather than the actual value for hints). Changing this to one larger than was actually used, with a minimum of two, significantly improved performance, to 27.7 seconds, or a reduction in processing time of about one-third. Enabling the hints further reduced processing time to 16.7 seconds, only a third of the initial amount. Interestingly, storing f_{1t} to use as a hint in subsequent iterations actually *increased* the execution time by a marginal amount, to 16.9 seconds.

²This is not true in the case of overdominance, which is not considered here.

4.5.4 Adding discounting

To this point, the only concern has been the maximum amount of *progress* that can be made, and only the final generation has been considered. It is not difficult, however, to modify the work done to this point to allow for an infinite horizon.

Previously, the program considered the effect of the major gene only in the final generation, and added the gain in polygenic value from each generation creating a sum equal to the value in the final generation.

It is a property of the genetic model that changes in polygenic value are permanent. As such, a change in generation t increases all generations by the same amount. To calculate a discounted value, `bestVal()` need only be modified to apply the identity in [3.5] to the calculated value for $\Delta \bar{A}_{t+1}$, discount this value and that returned from `nextVal()`, add the current value of the major gene, and multiply both by the discount factor of [3.4]. This is accomplished by a simple `if/then` structure in `bestVal()`. To stay with a single code base, these actions are taken only if the variable `discount` has a non-zero value. The only other change required is to change the print routine for tentative solutions such that the present value of each generation and its future is displayed. The present value *of the choice made* is reported for each generation. This is not the same as the present value of the generation; the value of the current state is not included. By calculating in this manner, it is easier to compare the relative value of choices later when choosing whether or not to test.

4.5.5 Using mass selection as a default choice

While the simplistic approach of using a very long planning horizon and increasing until the final generation has no present value could work, it is not guaranteed to do so without significant programming effort, as the library routines used have an upper bound on the p_t for which inverses of distribution functions may be calculated. However, a better method exists. In the simple approach, the entire value of the added generation is an increase, though it is possibly offset by different actions in prior generations (the steps that are optimal for n and $n + 1$ generations are not the same, and thus the first n steps of the $n + 1$ generation solution are worth less than the n generation solution). A more efficient solution than ignoring generations after n is to switch to mass selection after n , and to calculate the present value in that manner.

The only further modification required is to change `bestVal()` such that in the final generation, it returns the present discounted value of future generations under mass selection. The solution for an infinite horizon can then be found by allowing the finite horizon algorithm to run until it finds no value

in adding an additional generation of selection.

4.5.6 The value of mass selection

Starting with

$$\begin{bmatrix} x_{1t} & -x_{2t} & & -\frac{a}{\sigma}h^2 \\ x_{1t} & & -x_{3t} & -2\frac{a}{\sigma}h^2 \\ & & \Phi(-x_{1t})p_t^2 + \Phi(-x_{2t})p_t(1-p_t) + \Phi(-x_{3t})(1-p_t)^2 & -Q \end{bmatrix} = 0 \quad (4.67)$$

yields

$$\Phi(-x_{1t})p_t^2 + \Phi\left(-x_{1t} + \frac{a}{\sigma}\right)p_t(1-p_t) + \Phi\left(-x_{1t} + 2\frac{a}{\sigma}\right)(1-p_t)^2 - Q \quad (4.68)$$

or

$$L = \Phi(-x_{1t})p_t^2 + \Phi(-x_{2t})p_t(1-p_t) + \Phi(-x_{3t})(1-p_t)^2 - Q$$

and

$$\begin{aligned} L' = & x_{1t}\phi^2(-x_{1t})p_t^2 + \left(x_{1t} - \frac{a}{\sigma}\right)\phi^2\left(-x_{1t} + \frac{a}{\sigma}\right)p_t(1-p_t) \\ & + \left(x_{1t} - 2\frac{a}{\sigma}\right)\phi^2\left(-x_{1t} + 2\frac{a}{\sigma}\right)(1-p_t)^2 \end{aligned} \quad (4.69)$$

These equations are passed to a numeric routine for solution; reducing to the single variable x_{1t} speeds calculation. The resultant values of \bar{G}_t are calculated and discounted until p_{t+1} is within the constant peps, which is chosen as the limit of resolution for gene frequency, of 1.0. p_t is then accepted as being equal to one, and the gain in polygenic value becomes fixed for all later generations; [3.7] is used to add the value of all future gain.

4.5.7 Testing costs

The results so far consider only the revenue from the breeding program, and not the associated costs. Realistically, a cost should be imposed when animals are tested for the gene, and the breeding program should continue only so long as the benefits exceed the cost, the benefit being the gain *in excess of* the gain from mass selection. The algorithm changes only slightly to handle this variation: the best possible breeding choice is still found, but its value is compared to the value of switching to mass selection. If it does not exceed mass selection by the testing cost c , the switch is made. Using a discount rate of 8% and a test cost of .1 with Dekkers' parameters, the transition to mass selection occurs after the seventh generation. Program output is included in Appendix C.

4.5.8 Changing the horizon

The largest computational cost is not in calculating the values of the states, but in preparing the bubblettes for this computation. Once it is known that states beyond a given generation are not used, there is no reason to continue calculating these states. Similarly, if a breeding program has not switched to mass selection, a longer program may be desirable. The control variable `smartShrinkGens` is added to handle this situation.

With `smartShrinkGens` set, if mass selection is not chosen in the final choice generation, the time horizon is increased by one. The solution of the current iteration, augmented by mass selection for the final generation, is taken as the center, and the next iteration is run with the same grain.

Conversely, if the switch to mass selection occurs before the final generation, the generation in which the switch occurs becomes the final generation. However, the grain is reduced, as the available states are a subset of the states already considered.

This final model can be expressed as

$$\begin{aligned} \max_{T, \{f_t: 0 \leq t < T-1\}} L &= \sum_{t=0}^{T-1} (1-r)^{t+1} \left[a(2p_{t+1}(f_t) - 1) + \frac{\Delta \bar{A}_{t+1}(f_t)}{r} - c \right] \\ &+ (1-r)^T \left[\frac{a}{r} + \frac{\Delta \bar{A}}{r^2} \right] \end{aligned} \quad (4.70)$$

the selection of a length T of the optimal breeding program, and the selection of f_t for all periods of the program so as to optimize present discounted value at the beginning of the program.

CHAPTER 5 THE GENETIC PROBLEM WITH DISEQUILIBRIUM

The model developed in [4.70] is incomplete in several ways. The first omission to be considered is that of *gametic phase disequilibrium* between the major gene and polygenes. While not considered to this point, the selection intensity on the polygenes is weaker for type BB than for Bb , which is in turn weaker than for bb . This is a result of the higher fractions selected from BB and Bb —an animal of type BB does not need as high a polygenic value as one of type bb to be selected.

As with the first model, the actual choice variables, f_t , will not be used. Rather than a single state variable and an internal optimization given that variable, a pair of state variables will now be used: p_t , and the difference between the average polygenic values associated with the B and b gametes, $\bar{A}_{B,t} - \bar{A}_{b,t}$.

5.1 Gametic phase disequilibrium

The single-dimensional approach ignores the effects of “gametic phase disequilibrium,” a well known consequence of selection [Falconer, page 202]. Under selection, the superior homozygotes, BB , are subject to a lesser selection intensity for polygenic effects than the other types [Dekkers 98]; and as a consequence, have a lower polygenic value. Rather than a single average polygenic value \bar{A}_t , now two values, $\bar{A}_{B,t}$ and $\bar{A}_{b,t}$ must be distinguished, reflecting average polygenic values for gametes carrying the B and b alleles, respectively. As each individual gets two gametes, one from each parent, the average polygenic values for BB , Bb , and bb are then, respectively, $2\bar{A}_{B,t}$, $\bar{A}_{B,t} + \bar{A}_{b,t}$, and $2\bar{A}_{b,t}$. The overall average polygenic value is then a weighted average,

$$\bar{A}_t = 2p_t\bar{A}_{B,t} + 2(1-p_t)\bar{A}_{b,t} \quad (5.1)$$

Similarly to the single-dimensional problem, these average values can be written as recursive equations:

$$\bar{A}_{B,t+1} = \frac{f_{1t}p_t^2(\bar{A}_{B,t} + \frac{1}{2}i_{1t}\sigma) + f_{2t}p_t(1-p_t)\frac{1}{2}(\bar{A}_{B,t} + \bar{A}_{b,t} + i_{2t}\sigma)}{f_{1t}p_t^2 + f_{2t}p_t(1-p_t)} \quad (5.2)$$

and

$$\bar{A}_{b,t+1} = \frac{f_{2t}p_t(1-p_t)\frac{1}{2}(\bar{A}_{B,t} + \bar{A}_{b,t} + i_{2t}\sigma) + f_{3t}(1-p_t)^2(\bar{A}_{b,t} + \frac{1}{2}i_{3t}\sigma)}{1 - f_{1t}p_t^2 - f_{2t}p_t(1-p_t)} \quad (5.3)$$

the first of which is equation 10 in [Dekkers 98]. Note that this usage differs from Dekkers' introduction of $W_{B,t} = p_t \bar{A}_{B,t}$ and $W_{b,t} = (1 - p_t) \bar{A}_{b,t}$. While this change was advantageous for the use of optimal control, it would introduce complications for the methods developed below. The raw polygenic values have the same units, which allows their difference to be defined, while the W are weighted to reflect their contributions to the overall average polygenic value.

5.2 Finding the state variables

As before, there are more choice variables than state variables, and non-linearity makes the actual choice variables f_{mt} impractical for dynamic programming. Two state variables are now needed, and the mechanism for maximizing [4.55] is now irrelevant. The choice of p_{t+1} and a single additional variable for period $t + 1$ will *exactly* define f_{mt} , as [4.16], [5.2], and [5.3] create three equations in the three unknowns f_{mt} .

The new fitness function for a given generation, comparable to [2.10] is now

$$\bar{G}_t = a(2p_t - 1) + 2p_t \bar{A}_{B,t} + 2(1 - p_t) \bar{A}_{b,t} \quad (5.4)$$

However, substituting [5.2] and [5.3] into [5.4] does not yield a simple result such as [4.18], in which the change in \bar{A} is easily isolated. Instead, a dependence upon two variables of the prior generation, $\bar{A}_{B,t}$ and $\bar{A}_{b,t}$, remains. In order to create a useful algorithm, it is necessary to completely isolate the effects of the past states and the choices made; it is necessary to have an expression such as [4.18] which describes the effect of the choice on the objective function.

One way of doing this is to find an expression for $\bar{A}_{B,t+1} - \bar{A}_{b,t+1}$ in terms of $\bar{A}_{B,t} - \bar{A}_{b,t}$, p_t , p_{t+1} , and f_t . It is useful to define polygenic disequilibrium

$$d_t \equiv \bar{A}_{B,t} - \bar{A}_{b,t} \quad (5.5)$$

Using the relation that

$$i = \frac{z}{f} \quad (5.6)$$

[Falconer, equation 11.5], noting that the denominator of [5.2] is equal to Qp_{t+1} , and rearranging terms.

$$\begin{aligned} \bar{A}_{B,t+1} &= \frac{f_{1t} p_t^2 \bar{A}_{B,t} + f_{2t} p_t (1 - p_t) \frac{1}{2} (\bar{A}_{B,t} + \bar{A}_{b,t}) + p_t^2 \frac{1}{2} z_{1t} \sigma + p_t (1 - p_t) \frac{1}{2} z_{2t} \sigma}{Qp_{t+1}} \\ &= \frac{[f_{1t} p_t^2 + f_{2t} p_t (1 - p_t)] \bar{A}_{B,t} - f_{2t} p_t (1 - p_t) \frac{1}{2} (\bar{A}_{B,t} - \bar{A}_{b,t}) + p_t^2 \frac{1}{2} z_{1t} \sigma + p_t (1 - p_t) \frac{1}{2} z_{2t} \sigma}{Qp_{t+1}} \end{aligned}$$

$$\begin{aligned}
&= \frac{Qp_{t+1}\bar{A}_{B,t} - f_{2t}p_t(1-p_t)\frac{1}{2}d_t + p_t^2\frac{1}{2}z_{1t}\sigma + p_t(1-p_t)\frac{1}{2}z_{2t}\sigma}{Qp_{t+1}} \\
&= \bar{A}_{B,t} - \frac{f_{2t}p_t(1-p_t)}{2Qp_{t+1}}d_t + \frac{p_t^2z_{1t} + p_t(1-p_t)z_{2t}}{2Qp_{t+1}}\sigma
\end{aligned} \tag{5.7}$$

similarly,

$$\begin{aligned}
\bar{A}_{b,t+1} &= \frac{f_{2t}p_t(1-p_t)\frac{1}{2}(\bar{A}_{B,t} + \bar{A}_{b,t}) + p_t(1-p_t)\frac{1}{2}z_{2t}\sigma + f_{3t}(1-p_t)^2\bar{A}_{b,t} + (1-p_t)^2\frac{1}{2}z_{3t}\sigma}{Q - Qp_{t+1}} \\
&= \frac{f_{2t}p_t(1-p_t)\frac{1}{2}(\bar{A}_{B,t} - \bar{A}_{b,t}) + [f_{3t}(1-p_t)^2 + f_{2t}p_t(1-p_t)]\bar{A}_{b,t}}{Q(1-p_{t+1})} \\
&\quad + \frac{p_t(1-p_t)\frac{1}{2}z_{2t} + (1-p_t)^2\frac{1}{2}z_{3t}\sigma}{Q(1-p_{t+1})} \\
&= \frac{[f_{3t}(1-p_t)^2 + f_{2t}p_t(1-p_t)]\bar{A}_{b,t} + \frac{1}{2}f_{2t}p_t(1-p_t)d_t}{Q(1-p_{t+1})} \\
&\quad + \frac{\frac{1}{2}p_t(1-p_t)z_{2t} + (1-p_t)^2z_{3t}\sigma}{Q(1-p_{t+1})} \\
&= \frac{[Q - f_{1t}p_t^2 - f_{2t}2p_t(1-p_t) + f_{2t}p_t(1-p_t)]\bar{A}_{b,t} + \frac{1}{2}f_{2t}p_t(1-p_t)d_t}{Q(1-p_{t+1})} \\
&\quad + \frac{\frac{1}{2}p_t(1-p_t)z_{2t} + (1-p_t)^2z_{3t}\sigma}{Q(1-p_{t+1})} \\
&= \bar{A}_{b,t} + \frac{1}{2}\frac{f_{2t}p_t(1-p_t)}{Q(1-p_{t+1})}d_t + \frac{1}{2}\frac{p_t(1-p_t)z_{2t} + (1-p_t)^2z_{3t}\sigma}{Q(1-p_{t+1})}
\end{aligned} \tag{5.8}$$

yielding

$$\begin{aligned}
d_{t+1} &\equiv \bar{A}_{B,t+1} - \bar{A}_{b,t+1} \\
&= \bar{A}_{B,t} - \frac{1}{2Q}\frac{f_{2t}p_t(1-p_t)}{p_{t+1}}d_t + \frac{1}{2Q}\frac{p_t^2z_{1t} + p_t(1-p_t)z_{2t}}{p_{t+1}}\sigma \\
&\quad - \bar{A}_{b,t} - \frac{1}{2Q}\frac{f_{2t}p_t(1-p_t)}{(1-p_{t+1})}d_t - \frac{1}{2Q}\frac{p_t(1-p_t)z_{2t} + (1-p_t)^2z_{3t}\sigma}{(1-p_{t+1})}\sigma \\
&= d_t - \frac{1}{2Q}\left[\frac{f_{2t}p_t(1-p_t)}{p_{t+1}} + \frac{f_{2t}p_t(1-p_t)}{(1-p_{t+1})}\right]d_t \\
&\quad + \frac{1}{2Q}\left[\frac{p_t^2z_{1t} + p_t(1-p_t)z_{2t}}{p_{t+1}} - \frac{p_t(1-p_t)z_{2t} + (1-p_t)^2z_{3t}\sigma}{(1-p_{t+1})}\right]\sigma \\
&= d_t - \frac{1}{2Q}f_{2t}p_t(1-p_t)\left[\frac{1-p_{t+1} + p_{t+1}}{p_{t+1}(1-p_{t+1})}\right]d_t \\
&\quad + \frac{1}{2Q}\left[\frac{p_t^2}{p_{t+1}}z_{1t} + p_t(1-p_t)\left[\frac{1}{p_{t+1}} - \frac{1}{1-p_{t+1}}\right]z_{2t} - \frac{(1-p_t)^2}{(1-p_{t+1})}z_{3t}\sigma\right] \\
&= d_t - \frac{1}{2Q}f_{2t}\frac{p_t(1-p_t)}{p_{t+1}(1-p_{t+1})}d_t \\
&\quad + \frac{1}{2Q}\left[\frac{p_t^2}{p_{t+1}}z_{1t} - \frac{p_t(1-p_t)}{p_{t+1}(1-p_{t+1})}z_{2t} - \frac{(1-p_t)^2}{(1-p_{t+1})}z_{3t}\sigma\right]
\end{aligned} \tag{5.9}$$

and

$$\bar{A}_{B,t+1} - \bar{A}_{B,t} = \frac{f_{2t}p_t(1-p_t)}{2Qp_{t+1}} (\bar{A}_{B,t} - \bar{A}_{b,t}) + \frac{f_{1t}p_t^2 i_{1t}\sigma + f_{2t}p_t(1-p_t) i_{2t}\sigma}{2Qp_{t+1}} \quad (5.10)$$

Before proceeding to $\bar{A}_{b,t+1}$, it is useful to derive

$$\begin{aligned} 1 - p_{t+1} &= 1 - \frac{1}{Q} \{f_{1t}p_t^2 + f_{2t}p_t(1-p_t)\} \\ &= \frac{Q - f_{1t}p_t^2 - f_{2t}p_t(1-p_t)}{Q} \\ &= \frac{f_{1t}p_t^2 + f_{2t}2p_t(1-p_t) + f_{3t}(1-p_t^2) - f_{1t}p_t^2 - f_{2t}p_t(1-p_t)}{Q} \\ &= \frac{1}{Q} \{f_{2t}p_t(1-p_t) + f_{3t}(1-p_t^2)\} \end{aligned} \quad (5.11)$$

$$\begin{aligned} Q(1-p_{t+1})\bar{A}_{b,t+1} &= f_{2t}p_t(1-p_t) \frac{1}{2} (\bar{A}_{B,t} + \bar{A}_{b,t} + i_{2t}\sigma) \\ &\quad + f_{3t}(1-p_t)^2 \left(\bar{A}_{b,t} + \frac{1}{2} i_{3t}\sigma \right) \\ &= [f_{2t}p_t(1-p_t) + f_{3t}(1-p_t^2)] \bar{A}_{b,t} + f_{2t}p_t(1-p_t) \frac{1}{2} (\bar{A}_{B,t} - \bar{A}_{b,t}) \\ &\quad + f_{2t}p_t(1-p_t) \frac{1}{2} i_{3t}\sigma + f_{3t}(1-p_t)^2 \frac{1}{2} i_{3t}\sigma \\ &= Q(1-p_{t+1})\bar{A}_{b,t} + f_{2t}p_t(1-p_t) \frac{1}{2} (\bar{A}_{B,t} - \bar{A}_{b,t}) \\ &\quad + f_{2t}p_t(1-p_t) \frac{1}{2} i_{3t}\sigma + f_{3t}(1-p_t)^2 \frac{1}{2} i_{3t}\sigma \end{aligned} \quad (5.12)$$

then

$$\bar{A}_{b,t+1} - \bar{A}_{b,t} = \frac{f_{2t}p_t(1-p_t)}{2Q(1-p_{t+1})} (\bar{A}_{B,t} - \bar{A}_{b,t}) + \frac{f_{2t}p_t(1-p_t) i_{2t}\sigma + f_{3t}(1-p_t)^2 i_{3t}\sigma}{2Q(1-p_{t+1})} \quad (5.13)$$

allowing the calculation,

$$\begin{aligned} \Delta d_{t+1} &= (\bar{A}_{B,t+1} - \bar{A}_{b,t+1}) - (\bar{A}_{B,t} - \bar{A}_{b,t}) \\ &= \frac{f_{2t}p_t(1-p_t)}{2Qp_{t+1}} (\bar{A}_{B,t} - \bar{A}_{b,t}) + \frac{f_{1t}p_t^2 i_{1t}\sigma + f_{2t}p_t(1-p_t) i_{2t}\sigma}{2Qp_{t+1}} \\ &\quad - \frac{f_{2t}p_t(1-p_t)}{2Q(1-p_{t+1})} d_t - \frac{f_{2t}p_t(1-p_t) i_{2t}\sigma + f_{3t}(1-p_t)^2 i_{3t}\sigma}{2Q(1-p_{t+1})} \\ &= \frac{f_{2t}p_t(1-p_t)}{2Qp_{t+1}(1-p_{t+1})} d_t \\ &\quad + \frac{p_t^2(1-p_{t+1}) z_{1t} + p_t(1-p_t) z_{2t} + (1-p_t)^2 p_{t+1} z_{3t}}{2Qp_{t+1}(1-p_{t+1})} \sigma \end{aligned} \quad (5.14)$$

Also note that [5.10] and [5.13] can be written as

$$\bar{A}_{B,T} = \bar{A}_{B,0} + \sum_{t=0}^{T-1} \frac{f_{2t}p_t(1-p_t)}{2Qp_{t+1}} (\bar{A}_{B,t} - \bar{A}_{b,t}) + \frac{f_{1t}p_t^2 i_{1t}\sigma + f_{2t}p_t(1-p_t) i_{2t}\sigma}{2Qp_{t+1}}$$

$$= \bar{A}_{B,0} + \sum_{t=0}^{T-1} \Delta \bar{A}_{B,t+1} \quad (5.15)$$

and

$$\bar{A}_{b,T} = \bar{A}_{b,0} + \sum_{t=0}^{T-1} \Delta \bar{A}_{b,t+1} \quad (5.16)$$

and

$$\bar{A}_{B,T} = \bar{A}_{B,0} + \sum_{t=0}^{T-1} \Delta \bar{A}_{B,t+1} \quad (5.17)$$

Which means that, as was true in the case of a single state variable, changes in the average polygenic value are permanent. However, as [5.4] depends upon p_t , the value of $\Delta \bar{A}_{B,t}$ is not the same in all future generations. making calculations such as [4.42], which calculates the present value of a change, impossible.

Instead, consider

$$\begin{aligned} \bar{A}_{t+1} - \bar{A}_t &= 2p_{t+1}\bar{A}_{B,t+1} + 2(1-p_{t+1})\bar{A}_{b,t+1} - 2p_t\bar{A}_{B,t} - 2(1-p_t)\bar{A}_{b,t} \\ &= 2\bar{A}_{b,t+1} + 2p_{t+1}(\bar{A}_{B,t+1} - \bar{A}_{b,t+1}) - 2\bar{A}_{b,t} - 2p_t(\bar{A}_{B,t} - \bar{A}_{b,t}) \\ &= 2(\bar{A}_{b,t+1} - \bar{A}_{b,t}) + 2p_{t+1}d_{t+1} - 2p_t d_t \\ &= 2\left(\frac{f_{2t}p_t(1-p_t)}{2Q(1-p_{t+1})}d_t + \frac{f_{2t}p_t(1-p_t)i_{2t}\sigma + f_{3t}(1-p_t)^2i_{3t}\sigma}{2Q(1-p_{t+1})}\right) \\ &\quad + 2p_{t+1}d_{t+1} - 2p_t d_t \\ &= \frac{f_{2t}p_t(1-p_t)[d_t + i_{2t}\sigma] + f_{3t}(1-p_t)^2i_{3t}\sigma}{Q(1-p_{t+1})} + 2p_{t+1}d_{t+1} - 2p_t d_t \\ &= \frac{p_t(1-p_t)[f_{2t}d_t + z_{2t}\sigma] + (1-p_t)^2z_{3t}\sigma}{Q(1-p_{t+1})} + 2p_{t+1}d_{t+1} - 2p_t d_t \end{aligned} \quad (5.18)$$

which is written entirely in terms of the state variables p_t and d_t , and fractions which are functions of these choice variables. Thus, [5.18] can be used to write the present value of the change in state as

$$pv = \frac{2(1-r)}{r} [\bar{A}_{t+1} - \bar{A}_t] \quad (5.19)$$

5.2.1 The state

The state at any time will be described by the pair (p_t, d_t) , where d_t is the disequilibrium at time t ,

$$d_t = \bar{A}_{B,t} - \bar{A}_{b,t} \quad (5.20)$$

Rather than calculating f_{1t} to maximize the polygenic gain, the successor states p_{t+1} and d_{t+1} are chosen, and the choice made fully dictates f_t . Equations [4.17], [5.14], and [4.21] constitute a system of three equations in the three unknowns f_t , which can be solved by a library routine. As a practical matter, [4.21] is used to eliminate f_{3t} from the system to reduce computation.

5.2.2 Finding the choice variables

Rather than relying upon library routines, a first order Newton-Raphson search is used to determine the choice variables f_t that accompany a change in state from (p_t, d_t) to (p_{t+1}, d_{t+1}) . The maximal and minimal values for f_{1t} and f_{2t} are calculated, and a second order search is started at the midpoint of possible values for f_{2t} .

5.2.3 Mass selection with gametic phase disequilibrium

When ignoring disequilibrium, all genotypes had the same polygenic distribution, and mass selection truncation points x_t were calculated by setting

$$x_{1t} + \frac{h^2 a}{\sigma} = x_{2t} = x_{3t} - \frac{h^2 a}{\sigma} \quad (4.44)$$

This is no longer the case under disequilibrium; the new rule is

$$x_{1t} + \frac{2\bar{A}_{B,t} - a}{\sigma} = x_{2t} + \frac{\bar{A}_{B,t} + \bar{A}_{b,t}}{\bar{\sigma}} = x_{3t} + \frac{2\bar{A}_{b,t} + a}{\sigma} \quad (5.21)$$

which can be rewritten as

$$x_{1t} + \left(\frac{2\bar{A}_{b,t} + 2d_t - a}{\sigma} \right) = x_{2t} + \left(\frac{2\bar{A}_{b,t} + d_t}{\sigma} \right) = x_{3t} + \left(\frac{2\bar{A}_{b,t} + a}{\sigma} \right) \quad (5.22)$$

or

$$x_{1t} + \left(\frac{d_t - a}{\sigma} \right) = x_{2t} = x_{3t} - \left(\frac{d_t - a}{\sigma} \right) \quad (5.23)$$

5.2.4 Translation from state to choice variables

While working with the state variables d and p is sufficient to describe the *behavior* of the system, it is not enough to quantitatively *evaluate* a path: the original f_t , x_t and z_t are needed in equations such as [5.18] to determine the value of the proposed state to the breeder. Equations [5.18], [5.9], and [4.17] are three equations in the three choice variables, determining them exactly. While it is not possible to write an explicit, closed form solution, reducing the equations to a single variable in a single unknown allows solving with a fast numeric routine.

As f_{1t} has already been eliminated from [5.18], and f_{3t} has generally been treated as an artifact of the choice of f_{1t} and f_{2t} , f_{2t} seems to be the natural choice. Manipulating [4.17] yields

$$f_{1t} = \frac{Qp_{t+1} - f_{2t}p_t(1 - p_t)}{p_t^2} \quad (5.24)$$

which can in turn be used in [4.21] to get

$$\begin{aligned}
f_{3t} &= \frac{Q - \frac{Qp_{t+1} - f_{2t}p_t(1-p_t)}{p_t^2} p_t^2 - 2p_t(1-p_t) f_{2t}}{(1-p_t)^2} \\
&= \frac{Q - Qp_{t+1} + f_{2t}p_t(1-p_t) - 2p_t(1-p_t) f_{2t}}{(1-p_t)^2} \\
&= \frac{Q(1-p_{t+1}) - p_t(1-p_t) f_{2t}}{(1-p_t)^2}
\end{aligned} \tag{5.25}$$

both of which are placed into [5.9],

$$\begin{aligned}
y &= d_t - d_{t+1} - \frac{1}{2Q} f_{2t} \frac{p_t(1-p_t)}{p_{t+1}(1-p_{t+1})} d_t \\
&+ \frac{1}{2Q} \left[\frac{p_t^2}{p_{t+1}} z \left(\frac{Qp_{t+1} - f_{2t}p_t(1-p_t)}{p_t^2} \right) + p_t(1-p_t) \left[\frac{1}{p_{t+1}} - \frac{1}{1-p_{t+1}} \right] z(f_{2t}) \right. \\
&\left. - \frac{(1-p_t)^2}{(1-p_{t+1})} z \left(\frac{Q(1-p_{t+1}) - p_t(1-p_t) f_{2t}}{(1-p_t)^2} \right) \right] \sigma
\end{aligned} \tag{5.26}$$

which has a derivative of

$$\begin{aligned}
\frac{\partial y}{\partial f_2} &= -\frac{1}{2Q} \frac{p_t(1-p_t)}{p_{t+1}(1-p_{t+1})} d_t \\
&+ \frac{1}{2Q} \left[\frac{p_t^2}{p_{t+1}} x \left(\frac{Qp_{t+1} - f_{2t}p_t(1-p_t)}{p_t^2} \right) \frac{-(1-p_t)}{p_t} \right. \\
&+ p_t(1-p_t) \left[\frac{1}{p_{t+1}} - \frac{1}{1-p_{t+1}} \right] x(f_{2t}) \\
&\left. - \frac{(1-p_t)^2}{(1-p_{t+1})} x \left(\frac{Q(1-p_{t+1}) - p_t(1-p_t) f_{2t}}{(1-p_t)^2} \right) \frac{-p_t}{1-p_t} \right] \sigma \\
&= -\frac{\sigma}{2Q} \frac{p_t(1-p_t)}{p_{t+1}(1-p_{t+1})} \left[\frac{d_t}{\sigma} + (1-p_{t+1}) x_1 - (1-2p_{t+1}) x(f_{2t}) - p_{t+1} x_3 \right] \\
&= \frac{1}{2Q} \frac{p_t(1-p_t)}{p_{t+1}(1-p_{t+1})} d_t \\
&- \frac{1}{2Q} \left[\frac{p_t^2}{p_{t+1}} x \left(\frac{Qp_{t+1} - f_{2t}p_t(1-p_t)}{p_t^2} \right) \frac{1-p_t}{p_t} \right. \\
&+ p_t(1-p_t) \left[\frac{1}{p_{t+1}} - \frac{1}{1-p_{t+1}} \right] x(f_{2t}) \\
&\left. - \frac{(1-p_t)^2}{(1-p_{t+1})} x \left(\frac{Q(1-p_{t+1}) - p_t(1-p_t) f_{2t}}{(1-p_t)^2} \right) \frac{p_t}{1-p_t} \right] \sigma \\
&= \frac{1}{2Q} \frac{p_t(1-p_t)}{p_{t+1}(1-p_{t+1})} d_t \\
&- \frac{1}{2Q} \left[\frac{p_t(1-p_t)}{p_{t+1}} x \left(\frac{Qp_{t+1} - f_{2t}p_t(1-p_t)}{p_t^2} \right) \right. \\
&\left. + p_t(1-p_t) \left[\frac{1}{p_{t+1}} - \frac{1}{1-p_{t+1}} \right] x(f_{2t}) \right]
\end{aligned}$$

$$\begin{aligned}
& -\frac{p_t(1-p_t)}{(1-p_{t+1})}x\left(\frac{Q(1-p_{t+1})-p_t(1-p_t)f_{2t}}{(1-p_t)^2}\right)\Big]\sigma \\
= & \frac{p_t(1-p_t)}{2Q}\sigma\left[\frac{\frac{d_t}{\sigma}}{p_{t+1}(1-p_{t+1})}-\frac{x\left(\frac{Qp_{t+1}-f_{2t}p_t(1-p_t)}{p_t^2}\right)}{p_{t+1}}\right. \\
& \left.-\left[\frac{1}{p_{t+1}}-\frac{1}{1-p_{t+1}}\right]x(f_{2t})+\frac{x\left(\frac{Q(1-p_{t+1})-p_t(1-p_t)f_{2t}}{(1-p_t)^2}\right)}{1-p_{t+1}}\right] \\
= & \frac{p_t(1-p_t)}{p_{t+1}(1-p_{t+1})}\frac{\sigma}{2Q}\left[\frac{d_t}{\sigma}-(1-p_{t+1})x\left(\frac{Qp_{t+1}-f_{2t}p_t(1-p_t)}{p_t^2}\right)\right. \\
& \left.-x(f_{2t})+p_{t+1}x\left(\frac{Q(1-p_{t+1})-p_t(1-p_t)f_{2t}}{(1-p_t)^2}\right)\right] \tag{5.27}
\end{aligned}$$

which can be used to solve for the vector f_t that yields the desired transition between states.

5.2.5 Bounds for the fractions that can be selected

The limits on values must also avoid the case where so much f_1 is used that f_3 must exceed one. To find this limit, set

$$f_{3t} = 1 \tag{5.28}$$

and substitute [5.24] into [4.21] to get

$$Q = \frac{Qp_{t+1}-f_{2t}p_t(1-p_t)}{p_t^2}p_t^2 + f_{2t}2p_t(1-p_t) + (1-p_t)^2 \tag{5.29}$$

$$Q(1-p_{t+1}) - (1-p_t)^2 = f_{2t}p_t(1-p_t) \tag{5.30}$$

$$f_{2min} = \frac{Q(1-p_{t+1}) - (1-p_t)^2}{p_t(1-p_t)} \tag{5.31}$$

$$\begin{aligned}
f_{1max} &= \frac{Qp_{t+1} - \frac{Q(1-p_{t+1})-(1-p_t)^2}{p_t(1-p_t)}p_t(1-p_t)}{p_t^2} \\
&= \frac{Qp_{t+1}}{p_t^2} - \frac{Q(1-p_{t+1}) - (1-p_t)^2}{p_t(1-p_t)} \\
&= \frac{Q(p_{t+1}(1-p_t) - p_t(1-p_{t+1})) - p_t(1-p_t)^2}{p_t^2(1-p_t)} \\
&= \frac{Q(p_{t+1} - p_t p_{t+1} - p_t + p_t^2) - p_t(1-p_t)^2}{p_t^2(1-p_t)} \tag{5.32}
\end{aligned}$$

and the case where f_3 would be less than zero:

$$Q = \frac{Qp_{t+1}-f_{2t}p_t(1-p_t)}{p_t^2}p_t^2 + f_{2t}2p_t(1-p_t) + 0 \tag{5.33}$$

$$Q(1-p_{t+1}) = f_{2t}p_t(1-p_t) \tag{5.34}$$

$$f_{2max} = \frac{Q(1-p_{t+1})}{p_t(1-p_t)} \tag{5.35}$$

An initial guess of

$$f_{2t}^0 = \frac{f_{2min} + f_{2max}}{2} \quad (5.36)$$

is made, and a Newton Raphson routine is used to find the value for f_t which sets the y of [5.26] equal to zero.

5.2.6 Changes in bubble structure from the one-dimensional model

In the one-dimensional model of Chapter 4, each generation of the breeding program had an assigned bubble. If a state selected in the final solution lay on the boundary of a bubble, the bubbles were re-centered about this tentative solution. The iteration was then repeated with the same grain. This approach has both practical and theoretical problems in multiple dimensions. The theoretical problem is relatively minor, as explained below: that a fully calculated state cannot be reused if chosen in a different generation than that for which it was first calculated, as the optimal choice depends upon the number of generations remaining in which choices can be made. The practical problem is far more serious: with one dimension, it was possible to cover all of state-space with an initial course grain for the first iteration. With even two-dimensions, the choice space is too large to consider such an approach (even if it is assumed that reasonable *a priori* bounds could be placed on the disequilibrium d_t). As such, it would be necessary to move the bubbles repeatedly until an initial starting point is found. Not only is there is no reason to believe that the state-spaces chosen in later generations will be relevant when the bounds are moved for earlier generations, the recursive nature of the problem (in which earlier choices define the choice-space for later generations) strongly suggests that the old states will not be relevant.

The search is not stopped at the boundaries of bubbles; instead new bubbles are created in accordance with the routine illustrated in Figure 3.12 and indexed.

An additional problem is created in solving the theoretical problem in the case where a fixed number of generations is used: both the optimal behavior from a state and the value of choosing that state are dependent upon the number of generations remaining. Unfortunately, it is necessary to use the fixed horizon to validate the algorithm, as this is the only problem for which known results exist. Accordingly, the generation for which a state was calculated is stored in its bubblette, and the bubblette recalculated if called for a different generation. This behavior will be removed for the infinite horizon, with the result that the infinite horizon problem is computationally less expensive to solve than the finite.

5.2.7 Plateaus in the state space

One of the difficulties of any numeric algorithm is handling plateaus in the objective function. This issue did not arise in the single-dimensional model, even for fifteen generations. However, it appears in models with as few as five generations, and is pronounced by eight generations, in the two-dimensional problem.

With four or fewer generations of breeding, the algorithm finds the same solutions as routines based upon Dekkers' equations. There is a slight difference for five; for eight, the choices made are noticeably different. To check the algorithm, the state selected after four generations of Dekkers' eight-generation solution was used as the starting point for the algorithm, leaving it to solve the final four. Not only did the algorithm choose the same final four steps as it originally selected, but it was revealed that the value of those steps was slightly *better* than those found by Dekkers, using iterative methods and the partial analytical solution. Output is included in Appendix C.

When discounting is incorporated, the plateaus become a less serious problem. Part of the plateau problem comes from accumulated rounding through successive choices. The intervening choices carry no weight in the objective function, and thus there is no reason for the algorithm to pick a "better" intermediate state that reaches the same final state. As an economic problem in discounted present value, however, it is not the optimal genetic progress, but the present value, that matters: no distinction is made between the worth of two solutions with the same present value.

5.2.8 Ridges in the state space.

The second common problem with numeric routines is ridges through the space of the objective function which may be missed or stepped over by the routine. The algorithm has been observed successfully handling such ridges in the two-dimensional case.

The algorithm works by making successfully better approximations in each iteration, more closely approximating a continuous search space with each iteration.. As such, the number of additional bubbles created should drop over time. However, for some planning horizons, it was found that after an initial drop in the number of bubbles consumed, computation slowed while large numbers of bubbles were used. Furthermore, the choices made changed significantly at these times, finally settling at points past those even considered on the prior iteration. Inspection of the individual steps taken and states evaluated showed that the algorithm was working properly: in the prior iteration, all neighbor states of the selected points were indeed inferior. The finer grain, however, allowed consideration of a point *between* those previously considered, which was not only superior, but led to a sequence of improvements

beyond the prior search-space.

5.2.9 Discounting and changing horizon in two-dimensions

The discounted models and those with infinite and changing horizons are not considered in the two-dimensional space. No new issues of interest are presented compared to the one dimensional case . These issues are therefore left for the four-dimensional case.

CHAPTER 6 FOUR DIMENSIONS: THE GENETIC PROBLEM WITH DIFFERENTIAL SELECTION

A basic assumption for the one-dimensional and two-dimensional cases was that equal numbers of males and females were used for breeding in each generation. In practice, however, it is possible, and usually desirable, to breed each male, or “sire,” to more than one female, or “dam.” To accommodate this approach, model changes are minor. The gametes from the selected males and the selected females are treated separately, as if each gender bred independently.

Previously, the fraction Q of the entire population was bred to produce the next generation. Instead, a proportion Q will be bred, but with separate Q^s and Q^d , representing the respective fractions of the *entire* population to be bred, and with the identity

$$\frac{Q^s + Q^d}{2} = Q \quad (6.1)$$

and noting the ratio

$$Q^r = \frac{Q^s}{Q^d} \quad (6.2)$$

of the quantities of males and females to be bred. Given this, [4.16] becomes

$$Q^d = f_{1t}^d p_t^2 + f_{2t}^d 2p_t(1 - p_t) + f_{3t}^d (1 - p_t^2) \quad (6.3)$$

$$Q^s = f_{1t}^s p_t^2 + f_{2t}^s 2p_t(1 - p_t) + f_{3t}^s (1 - p_t^2) \quad (6.4)$$

with the superscripts “ d ” and “ s ” indicating dams and sires, respectively

In the past, Q^r was identically one in all cases, and Q^s and Q^d were each equal to half of Q . Translating equations such as [5.2] to address the sire’s gametes will generally require replacing Q with $2Q^s 2Q^d$, and labeling the choice variables f_{mt} as f_{mt}^s , and state variables *for the next generation*, such as p_{t+1} with counterparts such as p_{t+1}^s .

The key to understanding the model is to observe that the allowed choices for selection among the two genders is completely independent of the choice made for the other gender: (p_{t+1}, d_{t+1}) pairs are chosen separately for each gender, representing not the offspring, but the *gametes* contributed by the

respective gender. The actual state of the next generation is the *average* of the states of the gametes contributed by the two genders. Nonetheless, the choice space is still four-dimensional, even though it consists of a pair of two-dimensional subspaces, and it is the full choice that determines the available choices in the next generation.

Care must be taken to handle the fact that there are now *four* rather than three types of creatures because the Bb hybrid is now distinguishable from the bB hybrid, as different selection intensity upon the genders results in different values for $A_{B,t+1}^s$ and $A_{B,t+1}^d$ and also for the inferior allele. However, there is no way to inspect the offspring to determine whether it is in fact Bb or bB , and it is thus necessary to combine the two for selection purposes. Note that this is distinct from the approach used by Dekkers in unpublished work, where the two groups were treated as distinct to achieve theoretical results. Additionally, the actual distribution of the heterozygotes is bimodal, being a nested distribution. The first distribution is a Bernoulli, choosing between Bb and bB , while the second is a normal distribution. The two normal distributions have the same variance but different means. For computational and mathematical simplicity, the nested distribution will be taken as a single normal distribution, with appropriate modeling of the multi-modal solution left for future research.¹

Due to the differences it will not be possible to directly compare the results of the four-dimensional case with the Dekkers results. However, the states should still be “reasonably” close to those of Dekkers, though Dekkers’ reported values should be superior, due to the ability to distinguish between the two types of heterozygotes. While such a distinction is of interest to theoretical quantitative genetics, it has no economic relevance, as it is impossible to distinguish between the two groups of heterozygotes in an actual animal herd unless full information on the parentage of each animal is available, and at least one of the parents was a homozygote of known type. Additionally, it is still possible to perform a check on the algorithm by setting the number of males and females equal; with this setting, it should achieve the same results as the two-dimensional algorithm, albeit more slowly.

6.1 Changing the model

Given that the genders are treated independently, producing gametes which are then combined with those of the other gender, changing the prior equations takes nothing more than compensating for the different fraction of the population used and introducing superscripts for some of the variables. Equations [5.2] and [5.3] then become

¹In fact, it is only bimodal after the initial round of selection, with the multi-modality increasing geometrically. The question would be how to accurately model the nested distribution as a single distribution.

$$\bar{A}_{B,t+1}^s = \frac{f_{1t}^s p_t^2 (\bar{A}_{B,t} + \frac{1}{2} i_{1t}^s \sigma) + f_{2t}^s p_t (1-p_t) \frac{1}{2} (\bar{A}_{B,t} + \bar{A}_{b,t} + i_{2t}^s \sigma)}{f_{1t}^s p_t^2 + f_{2t}^s p_t (1-p_t)} \quad (6.5)$$

and

$$\bar{A}_{b,t+1}^s = \frac{f_{2t}^s p_t (1-p_t) \frac{1}{2} (\bar{A}_{B,t} + \bar{A}_{b,t} + i_{2t}^s \sigma) + f_{3t}^s (1-p_t)^2 (\bar{A}_{b,t} + \frac{1}{2} i_{3t}^s \sigma)}{1 - f_{1t}^s p_t^2 - f_{2t}^s p_t (1-p_t)} \quad (6.6)$$

equation [4.17] becomes

$$p_{t+1}^s = \frac{1}{Q^s} \{ f_{1t}^s p_t^2 + f_{2t}^s p_t (1-p_t) \} \quad (6.7)$$

The sires' disequilibrium also remains similar,

$$\begin{aligned} d_{t+1}^s &= d_t - \frac{1}{4Q^s} f_{2t}^s \frac{p_t (1-p_t)}{p_{t+1}^s (1-p_{t+1}^s)} d_t \\ &\quad + \frac{1}{4Q^s} \left[\frac{p_t^2}{p_{t+1}^s} z_{1t}^s - \frac{p_t (1-p_t)}{p_{t+1}^s (1-p_{t+1}^s)} z_{2t} - \frac{(1-p_t)^2}{(1-p_{t+1}^s)} z_{3t} \right] \sigma \end{aligned} \quad (6.8)$$

with the equations for the dams being identical except for the new superscripts.

Selection will typically be more intense among sires than dams, since fewer need be chosen, and the increases in both frequency and disequilibrium will be more pronounced among the sires. The two are averaged (each animal has one parent of each gender, regardless of the proportion of sires and dams) to find the values for the entire population and the progeny:

$$\begin{aligned} d_{t+1} &\equiv \bar{A}_{B,t+1} - \bar{A}_{b,t+1} \\ &= \frac{1}{2} (\bar{A}_{B,t+1}^s + \bar{A}_{B,t+1}^d) - \frac{1}{2} (\bar{A}_{b,t+1}^s + \bar{A}_{b,t+1}^d) \\ &= \frac{1}{2} (\bar{A}_{B,t+1}^s - \bar{A}_{b,t+1}^s) + \frac{1}{2} (\bar{A}_{B,t+1}^d - \bar{A}_{b,t+1}^d) \\ &= \frac{1}{2} (d_{t+1}^s + d_{t+1}^d) \end{aligned} \quad (6.9)$$

frequency directly averages as well,

$$p_{t+1} = \frac{1}{2} (p_{t+1}^s + p_{t+1}^d) \quad (6.10)$$

and the transition rule for \bar{A}_{t+1} remains unchanged from [5.18]

$$\bar{A}_{t+1}^s - \bar{A}_t = \frac{p_t (1-p_t) [f_{2t}^s d_t + z_{2t}^s \sigma] + (1-p_t)^2 z_{3t}^s \sigma}{Q^s (1-p_{t+1}^s)} + 2p_{t+1}^s d_{t+1}^s - 2p_t d_t \quad (6.11)$$

and

$$\bar{A}_{t+1} - \bar{A}_t = \frac{\bar{A}_{t+1}^s - \bar{A}_t}{2} + \frac{\bar{A}_{t+1}^d - \bar{A}_t}{2} \quad (6.12)$$

6.2 Changing the pattern of search

As before, a state will be accepted as an optimum if it is superior to all of its neighbors, and progress is made by examining all neighbor-states. With two-dimensions, there were only $3^2 - 1$, or eight, neighbors for each state, and no particular attention was paid to the order of the search. However, with four dimensions, there are $3^4 - 1$, or eighty, neighbors for each state. increasing the dividends for giving careful thought to the order of search. Each step in a direction that can be ruled out as unlikely prior to computation saves the wasting of not only the step itself, but more importantly it avoids the calculation of a cascading stream of futile searches with potentially 81^i searches in the generation i generations after the present.

Working from known theoretical results from Dekkers' unpublished research, it is to be expected that in most cases, selection on the major gene will be more intense than under mass selection—that is, that the gene frequency will increase at a more rapid rate for the dams, and even more rapidly for the sires. As such, the search is reordered so that an increase in both frequencies is the first state considered, followed by an increase for the sire and none for the dam, followed by a search only for the dam.

In the same vein, to avoid the other seventy-seven steps and the steps descending from those steps, the check for the possibility of increasing frequency is made immediately upon the initial step: if the state with the default change in disequilibrium is not a valid transition, the nearest disequilibrium for the selected frequency is immediately checked. Furthermore, if the state is not chosen as a step, the information is saved for the individual checks for the sires and dams alone.

The result of the two changes is a search that, considering four generations, solves its initial iteration in less than a minute rather than taking several minutes without the reordered search, again using one processor of a dual Pentium 333..

6.3 Adding a cache

The four-dimensional state is separable in the sense of [3.9], in that the states chosen for males and females are independent of each other—while the states chosen will be combined to form the next generation, the state chosen for the sires in no way constrains the state for the dams, and vice versa. As such, for any sub-state or sub-choice (p_{t+1}^s, d_{t+1}^s) , the value of $\Delta \bar{A}_{t+1}^s$ is the same, regardless of the choice made for the dams. Similarly, the value of ΔA_{t+1}^d is independent of the choice made for the sires. Note, however, that as Q_s and Q_d are distinct, knowing ΔA_{t+1}^s for a successor subs-state gives no

information about ΔA_{t+1}^d , and vice-versa. Also recall that these values are dependent upon the current state and reflect the gain achieved in moving to the successor state. Accordingly, while there is value in temporarily caching these values, there is no gain in saving the various values once a successor state is chosen, as they are unique to that particular present state.

6.4 Partial additivity

To this point, all genes considered have been “fully additive.” That is, having a gene twice is exactly twice as good as having it once. Many genes of interest, however, are only partially additive, with two copies superior to a single copy, but not of twice the value. Such genes require only a simple change to the model, with the contribution from the major gene changing from

$$G_{major} = (2p_t - 1) a \quad (6.13)$$

as in [2.10], to

$$G_{major} = (p_t^s + p_t^d - 1) a + (p_t^s + p_t^d - 2p_t^s p_t^d) d \quad (6.14)$$

where d is a non-centrality parameter, and the three types are valued as $-a$, d , and a , respectively, for zero, one, or two copies. The only changes required are in the evaluation of fitness, not the algorithm itself.

CHAPTER 7 OPTIMAL BREEDING WITH THE ESTROGEN RECEPTOR LOCUS

Advances in biotechnology in recent years have permitted the identification of specific genes in animals, and the quantification of the effects of these genes. A gene of particular interest to swine producers is the *Estrogen Receptor Gene*, (ESR) which, among other effects, increases litter size [Rothschild 94]. Litter size is of direct economic impact, as it allows the producer larger output from the same breeding stock, the same output from fewer breeding animals, or a combination of the two. Furthermore, litter size is a trait with poor heritability; it is a difficult trait to enhance by conventional methods, as it has poor heritability; very little of the observed value is passed from a sow to her daughters.

Determining whether or not the animal has the desirable allele of the gene and how many copies of that allele is not without cost. To do so necessitates that each animal be tested, incurring both a direct cost of testing and a royalty to the developer of the test. It is not clear *a priori* that testing is always of value. The analysis which follows is an application using the ESR. Throughout the analysis, an assumption of a test cost of three dollars will be used, along with a royalty from fifteen to twenty dollars. Both figures are drawn from private correspondence with Max Rothschild of Iowa State University.

The following will assume an enterprise with a breeding herd kept at a constant size, with all pigs born either kept to breed within the herd or sold as feeder pigs. Financial gains from selection would be significantly greater if the sows just below the cutoff point for breeding were sold as breeding stock, but this is left for future research. Finally, some of the initial examples assume completely random mating for simplicity, but this is noted when used.

Assuming a total cost of twenty dollars for the test, an average litter size of ten, an existing gene frequency of one-half, a marginal economic value of thirteen and one-half dollars [Guidelines] per additional live pig, and an additive effect of .31 additional pigs born live drawn from unpublished research of Short, *et al* [Short], choosing a sow with the gene two copies of the gene has an expected return of

$$2 \times .31 \times \$13.50 = \$8.38 \quad (7.1)$$

as compared to a sow with no copies, approximately one-half of the cost of the test. On the other hand,

if each sire is bred five times, it will have the same number of offspring with or without the ESR, as only sows have litters, but will pass the gene to five daughters that breed, on average, for a return after two generations (when the sires daughters breed) of

$$2 \times 5 \times .31 \times \$13.50 = \$41.90 \quad (7.2)$$

slightly exceeding the cost of two tests. However, if multiple sires must be tested to choose one with the gene, the benefit is then outweighed by the cost.

Nonetheless, it may still be worthwhile to test. It is not only the present generation, but each successive generation that is improved. Thus with a generation period of eighteen months, an annual discount rate of 8% for a generational discount rate of approximately 12%, with each sow breeding only once with an average litter size of ten, a selected homozygote sire expects, on average, to have five daughters (one from each sow) and one-half son (one-tenth from each sow) breed. The daughters will pass on the sire's gene one-half the time to their own breeding daughters and to one-tenth a breeding son each, while the sons pass on to five daughters and another half son, indefinitely. The result is an additional five copies of the gene among the sows that breed in each generation from the second generation onward.

Considering the sire as generation zero, the additional present value of a homozygous sire with the ESR, as compared to a homozygous sire without the gene, is

$$\begin{aligned} pv &= \frac{5 \times .31 \times \$13.50}{(1+r)^2} \sum_{i=0}^{\infty} \frac{1}{(1+r)^i} \\ &= \frac{\$20.925}{r(1+r)^2} \\ &= \$139 \end{aligned} \quad (7.3)$$

a significant gain. Note that this understates the gain, as the one-half of a breeding son and five breeding daughters come from the assumption that that animals are randomly selected to breed; as the offspring will have higher estimated breeding values on average than those from sires without the ESR, the descendants of sires with the gene will be more likely to breed than others in the herd, and the actual number of breeding offspring will be larger.

Finally, the gene frequency should be considered. If the gene is initially present in half the population, then, designating the presence of the gene as B and the absence as b , it is to be expected that one in every four is a superior homozygote with the gene twice (BB), and one in every two is a homozygote with the gene once (Bb and bB), and that one in every four is an inferior homozygote (bb). This would mean an average of eight tests—four from the sires that would otherwise breed and four from those

who would not—at a total cost of \$160 to find an inferior homozygous sire to replace with a superior homozygous sire. Below this frequency, it will be necessary to test more from the *not to be bred*, and less from the *to be bred* pools to find a pair of sires to exchange, and vice versa when above this frequency. Eight is the *minimum* expected number of tests to make a selection when drawing at random—at a gene frequency of 50%, there is a one-fourth chance of finding the desired type on any draw, requiring an expected four draws each from the pools tentatively selected to be bred and not bred. As the frequency increases past 50%, the number that must be drawn to find an inferior homozygote increases faster than the number that must be drawn to find a superior homozygote decreases.

Formally, the expected number that must be tested to find a superior homozygote among sires that would not otherwise breed is

$$n_{BB}^e = \frac{1}{p^2} \quad (7.4)$$

while the expected number to find an inferior homozygote that would otherwise breed is

$$n_{bb}^e = \frac{1}{(1-p)^2} \quad (7.5)$$

The minimum can be found by differentiating the sum and setting the derivative to zero:

$$0 = \frac{-2}{p_{min}^{-3}} + \frac{2}{(1-p_{min})^{-3}} \quad (7.6)$$

or that

$$p_{min} = 1 - p_{min} \quad (7.7)$$

thus

$$p_{min} = .5 \quad (7.8)$$

Note that Equation [7.3] shows that with the original assumptions, the expected return of \$139 will not cover the expected testing cost of \$160. However, if each sow has multiple litters, fewer sires are needed and it is possible that the cost may be covered or exceeded—two sows per sire yields a return of \$278, easily covering the cost. Additionally, the \$139 figure is understated.

On the other hand, the testing of sows can never cover the cost. As described above, each sow has on average one daughter who breeds, and this daughter will inherit half of the mother's genetic material. Similarly, the daughter will produce on average one granddaughter to the original sow with one fourth of the original genetic material, and so fourth. Even without discounting, this means that replacing an inferior homozygous dam with a superior homozygote yields an expected total of two breeding sows and a total of four extra copies of the superior gene. Even assuming that all of these occur in a manner such that they result in a superior homozygote breeding instead of an inferior homozygote, this results

in only .62 additional pigs born alive, for a value of \$7.80. To cover the minimal cost of \$160, each sow would have to have more than twenty litters just to cover the cost of testing..

Given these results, it would tentatively appear that only the sires should be tested, as the dams do not spread the gene to enough dependents to justify the cost of testing.

7.1 Estimated breeding value and the ESR

In the previous chapters, it was assumed that the estimated breeding value (EBV) could be easily observed from phenotypic traits of the animal. With the ESR, however, it is by definition impossible to *observe* the trait in the animal until *after* breeding, because the phenotypic trait of interest is the number born alive (NBA). With dams, it is at least possible to make an observation after a litter is born. With sires, however, it is not possible to make the direct observation until their female offspring breed.

Fortunately, there are other methods by which an EBV can be obtained. While the animal has provided no direct information, its parents, grandparents, older siblings, and their relatives do yield information. By combining these, it is possible to make an estimate for an unbred animal. In unpublished work, Dekkers has found that such estimates have an accuracy of .32, yielding an effective heritability h^2 of .32² or .1024. This figure will be used throughout.

7.2 Choosing which animals to breed with truncation selection

The optimal fraction of each group to breed can be found from the algorithm in the prior chapter and is independent of the cost of testing. Given the truncation points for the three groups, animals may be divided by estimated breeding value into three categories: those who will reproduce regardless of genotype, those for whom the genotype will be required, and those who will not breed regardless of genotype. There is no point in testing the first and third groups.

While the genotype of individual animals cannot be known without paying for testing, it is assumed that the overall gene frequency of the initial population is known ahead of time. This knowledge, along with the choices made, whether optimal or mass selection, allow the calculation of the overall gene frequency in all subsequent generations.

The truncation point on EBV for optimal selection will be lowest for the homozygote with the gene, and highest for the homozygote without. Letting x_{mt} represent the truncation points for the three types, and f_{mt} the corresponding fractions selected, only animals that would qualify if of the first genotype,

but not if of the third, need to be tested. As optimal selection selects superior homozygotes that would be rejected with mass selection and rejects inferior homozygotes that would have been selected with mass selection, the animals that must be tested are those whose observable values are between these truncation points. Recalling that

$$f_{mt} = 1 - \Phi^{-1}(x_{mt}) \quad (7.9)$$

or

$$x_{mt} = \phi(1 - f_{mt}) \quad (7.10)$$

and that

$$d_t \equiv \bar{A}_{B,t} - \bar{A}_{b,t} \quad (7.11)$$

relationships can be written for animals of one type that are at the truncation point for another type. For example, for a superior homozygote with the same EBV as an inferior homozygote at the truncation point, the relation

$$x_{1't} + \frac{2A_{B,t} + a}{\sigma} = x_{3t} + \frac{2A_{b,t} - a}{\sigma} \quad (7.12)$$

holds, yielding

$$\begin{aligned} x_{1't} &= x_{3t} - \frac{2A_{B,t} - 2A_{b,t} + 2a}{\sigma} \\ &= x_{3t} - 2\frac{d_t + a}{\sigma} \end{aligned} \quad (7.13)$$

as the truncation point for type one animals that must be tested. The portion of these animals which must be tested is the difference between the fractions corresponding to the optimal truncation point and the point in [7.13]. Writing such relationships for each of the three types, namely the difference between the fractions that meet the EBV cutoffs for superior and inferior homozygotes, and multiplying by the respective portions of the total population represented by each type, it can be observed by substituting [7.13] and the corresponding equation for the heterozygote into [7.9], and the resulting expressions into the population constraint [4.16] that the fraction of the male population that must be tested is

$$\begin{aligned} f_t^{test} &= \left\{ f_{1t} - \left[1 - \Phi^{-1} \left(x_{3t} - 2\frac{d_t + a}{\sigma} \right) \right] \right\} p_t^2 \\ &+ \left\{ \left[1 - \Phi^{-1} \left(x_{1t} + \frac{d_t + a - d}{\sigma} \right) \right] - \left[1 - \Phi^{-1} \left(x_{3t} - \frac{d_t - a - d}{\sigma} \right) \right] \right\} 2p_t(1 - p_t) \\ &+ \left\{ \left[1 - \Phi^{-1} \left(x_{1t} + 2\frac{d_t + a}{\sigma} \right) \right] - f_{3t} \right\} (1 - p_t^2) \end{aligned} \quad (7.14)$$

where the f_{mt} and x_{mt} are the optimal values.

7.3 Assumptions and parameter values

Relying on the values estimated by Short, *et al* [Short], and those provided by the National Swine Improvement Federation [Guidelines], a weighted estimate of the value of the gene will be used. Each dam will be assumed to have three litters, though all three will be treated as occurring at the time of her second litter for purposes of discounting. The model becomes too cumbersome if the litters are treated as separate events, while not providing any additional information. Furthermore, the three litters can be taken as a single large litter, distributed with a mean equal to the sum of the three means, and a standard deviation calculated by assuming that the three litters are independent. This slightly overstates the actual variance, but by a very small amount for the given values of h^2 . Table [7.1] reproduces values from Tables 1 and 2 in the aforementioned paper, as well as the weighted averages.

The base interest rate and generation times represent a consensus value from experts in the Department of Economics, at Iowa State University, and are 8% and eighteen months, respectively. These values are also reproduced in Table 7.1.

7.4 Modification of the algorithm

The four-dimensional algorithm of the previous chapter was designed with the assumption that optimization would be performed for both sexes. It is already known, however, that the testing of dams can never cover its cost. Accordingly, the calculations relating to dams are disabled, and dams are always chosen by mass selection. It is also necessary to modify the search order to accommodate this change, as only the first two-dimensions are searched. Additionally, as optimization of the herd for a single trait precludes consideration of other traits, the breeding program is limited to five generations, with the improvements gained by the fifth generation considered permanent, and kept for all future generations.

7.5 Results

Using the output included in Appendix C, a five generation breeding program with mass selection is found to have a present value of \$554.07 per breeding sow in the herd. Optimal selection improves this to \$586.21, an improvement of \$32.14.

The algorithm was run again with a one percent increase in the interest rate (to 18.08%), with the

Table 7.1 Estimated parameters of the ESR gene

Expression	First Parity	Second parity	Value
Additive	.39	.31	.1337
Dominance	.05	.14	.11
NBA	9.2	10.3	9.93
σ_{NBA}	3.4	3.5	3.47
weight	1.	2.	
heritability of selection index			.1024
Initial frequency			.5
Sires Selected			1.34%
Dams selected			6.7%
Discount rate			8.0%
Generation time			18 months
Discount/gen			12.0%

Table 7.2 Results from optimal breeding of the ESR gene

Interest Rate	Mass Selection Value	Optimal Selection Value	Difference	Improvement
18.00%	\$554.07	\$586.21	\$32.14	5.80%
18.18%	\$546.56	\$578.21	\$31.65	5.89%

results are shown in Table 7.2. The interest rate elasticity of gains from production can be estimated as

$$\begin{aligned}\epsilon &= \frac{\$31.65 - \$32.14}{\$31.65} \times 100\% \\ &= -1.55\end{aligned}\tag{7.15}$$

CHAPTER 8 SUMMARY AND CONCLUSIONS

8.1 Analytic methods

The problem of maximizing genetic progress over a finite number of generations for a quantitative trait with an identifiable Quantitative Trait Locus or major gene has previously been shown to have a partial analytic solution [Dekkers 98]. The problem has control equations moving both forward and backwards in time, and the solution requires the value of one of the Lagrangian multipliers for the final generation, which is known from the actual parameter that the shadow value represents. Having set this parameter, the problem can be solved recursively by standard iterative computational methods.

Any problem beyond maximizing the value of the final generation becomes an economic problem. Particularly, any question other than “what is the best that can be achieved for this generation” must involve two or more generations of animals, which in turn requires a choice as to how to weight the response achieved in each generation, an economic question.

Of particular economic interest is the problem with an infinite planning horizon, in which the present discounted value of all future generations is considered. The limiting value of the Lagrangian multiplier for the infinite horizon model is known, and is in fact the same as in the finite. Unfortunately, as seen in §4.2, it is not possible to recurse backwards in time from this value while the other equations move forwards in time. If the value of any of the multipliers were known in any period, all others could be calculated, but four multipliers appear in the state equation for any generation, three from that period, and one from the next. For any finite number of periods, the equations contain exactly one multiplier too many, and the infinite horizon problem cannot be partially solved analytically as the finite horizon model was.

8.2 Dynamic programming and bubbles

The literature is completely devoid of optimal control and dynamic programming solutions for recursive equations in continuous variables that cannot be approximated as linear-quadratic. The

genetic problem, however, is a high order multinomial—for n periods, it has a component of order $2n$ in $2n$ variables, as well as components containing the standard normal density of functions of the inverse of the standard normal cumulative distribution function.

There is a body of dynamic programming literature which discretizes the problem and applies successively more precise approximations to successively smaller search spaces. However, these methods require advance knowledge of the search space so that it can be divided, and consider contiguous search spaces, which are generally compact or transformations of compact spaces. Not only do the valid search spaces for the genetic problem not meet these criteria, but the search space in each generation is *determined by* the choice in the prior generation. Furthermore, allocating computer storage even for pointers to the entire search space is beyond the memory capacity of current machines.

The entire search space is not of interest, however. In fact, only a very small subset of the total space is of interest for any given generation. A new method of dynamic programming is developed herein that creates subsets of the total space, or “bubbles.” These bubbles take the form of hypercubes whose dimension is the number of independent state variables in each stage. Each of these bubbles is divided into “bubblettes,” each of which represents a potential state.

An initial trial solution is selected, and bubbles are centered around the states chosen for each generation or stage. The problem is then solved recursively by querying the starting state for its value. This state in turn queries the value of the first step of the trial solution, and all states or bubblettes adjacent for their values, which in turn query the later generations. When values better than the tentative solution are found, the corresponding bubblette becomes the tentative solution, and the search begins again about that bubblette. When the initial query is finally returned, the grain, or distance between states, is halved, and the algorithm is run again, using the prior result as the starting point.

Once a bubblette is calculated, its value is stored, so that it (and its successors) need not be recalculated if queried again from a different predecessor bubblette. The value of a bubblette is the value of the successor it chooses, discounted if appropriate, and the value gained in making the transition to that state. That is, the bubblette in the next stage with the highest value is not necessarily the one chosen, as another bubblette less valuable in its own right may have a greater transitional gain.

The animal breeding problem is solved by turning the “real” choice variables, the fraction of each major gene type selected, into functions of the state transitions. The major gene frequency and the level of gametic phase disequilibrium are the state variables used, while the polygenic breeding value is taken as a consequence of the choices made. This is done by making the *change* in the polygenic

breeding value the transitional value between generations, allowing its calculation at any time as the initial value plus the changes along path to the current state.

The infinite horizon problem is solved by considering that there is a finite cost of testing the animals, halting the algorithm when the cost of testing exceeds the present value of its returns. The returns from testing are the difference between the present value of the optimal path and the present value of making the default choice, mass selection.

Using the values polygenic and major gene values from Dekkers' work, the algorithm successfully matched the theoretical results in one dimension (disequilibrium ignored) and two dimensions per generation for fifteen generations (the longest horizon considered) and for four dimensions per generation for five generations, the longest horizon considered. Optimal selection shows minimal gain over mass selection for longer periods. The algorithm successfully traversed ridges and navigated plateaus in state-space, the two primary difficulties faced in numerical algorithms.

Duplicating existing results with the algorithm is primarily of interest to verify that the algorithm performs properly. Having matched these results, the algorithm was also used to find the maximal values for the discounted one-dimensional model, and the infinite horizon one-dimensional model. The one-dimensional model was used as an intermediate step in the development of the full methods, as it is substantially faster yet contains all of the elements of the multi-dimensional model; only the function for the value of states changes. With the algorithm complete, it was used to evaluate an actual gene, the Estrogen Receptor Gene, or ESR, in swine.

8.3 The cache used for the bubbles

One of the problems addressed is that it is not known in advance which portions of the search space will be used prior to running the algorithm—or even before running an iteration within the algorithm. Another is that it is entirely possible that the portions considered will not fit into memory. Finally, as new bubbles are created, a method of finding them if used later is required.

The problems are simultaneously solved by using a cache with an indexing system, which is the most important contribution of this method. The bubbles are hypercubes, and their centers are placed upon predefined points (with an exception for the trial solution). This allows computationally cheap integer arithmetic to calculate where the center of the bubble housing any state *would be* if the bubble existed.

Many times when a state is needed, there is an initial guess as to which bubble holds that state's bubblette, assuming that it exists. One possibility is that it is in the initial trial solution for the next generation, while another is that it is in the same bubble as an already-referenced bubblette. In fact,

the latter case is the reason for indexing bubbles instead of bubblettes—since a state is identified as “best” by having a better value than all of its adjacent states, there is a high probability that the neighbors of a bubblette will be referenced immediately after that bubblette. By passing the bubble of the neighbor bubblette just referenced to the indexing routine as a hint, an expensive search can frequently be avoided.

While such hints will frequently avoid an expensive search, this is not always the case. To search the cache of bubbles, the index is first calculated, and the index of bubble centers checked for a match. This can be done by a sequential check, but it is faster to use intrinsic functions to find the location of the minimal value of the absolute value of the difference between the index array and the target value. With either method, if a match is found, its location is returned and the routine exits.

If the bubble is not in the cache, it must be created. If any unallocated bubbles remain available, the new bubble is simply created, and the location returned. If the cache is full, however, the least-recently referenced bubble is removed, and its location given to the new bubble. This is accomplished by giving each instance of the query function a successively greater number and storing this number for the bubble whenever it is referenced. This increases the chance, but does not guarantee, that bubbles in a superior search region will be available if needed again.

Finally, it should be noted that while the cache is generally considered as a single entity available for all generations, there appear to be significant performance gains for the genetic problem in reserving sections of the cache for each generation. This is because a large search in a later generation has the potential of removing all of the values calculated for earlier generations, requiring the recursive search to be repeated. However, this type of segregation of the cache may not always be available and may actually slow the search if the sections allocated to individual generations are too small.

8.4 The ESR gene

The algorithm found that with an 8% annual interest rate, the improvements from optimal selection are nearly 6% greater than those from mass selection. This result is based upon the assumption that the offspring are sold as feeder pigs. This improvement roughly covers the cost of testing for the major gene, and is not an economically viable solution. Selling a portion as breeding stock would substantially increase the value of gains from selection, and likely cover the costs of testing.

8.5 Conclusions for the dynamic programming method

New classes of problems are open to computational solution. Particularly, unimodal multivariate functions may be optimized over large search spaces in the absence of analytical solutions. The unimodality requirement could be relaxed if the search space were partitioned prior to search. Of particular interest are recursive problems with long horizons, such as the genetic breeding problem, macroeconomic optimal growth problems, and repeated games with continuous choice variables.

The curse of dimensionality is pushed back within this class of cases. Rather than considering the entire search space, noncontiguous regions within the space are indexed, avoiding the computational cost of indexing individual states—and more importantly, of conducting a search through such states..

The primary advance from this research comes from the observation that while only a tiny fraction of the search space will be examined, the states near an examined point are also likely to be examined. Constructing bubbles with adjacent states, and implementing a workable search algorithm so that these bubbles can be found, makes the search through previously calculated states practical.

Rather than ten billion possible states, a genetic problem in four dimensions taking fifteen generations requires as little as 15^{55} or less than 50,000 states, and may be calculated in an afternoon on a 333 MHz Pentium II. This figure is for an arbitrarily fine grain.

Finally, although more complicated problems can be solved by this method than by older dynamic programming methods, the number of dimensions will remain the limiting factor—not due to memory limitations, as in the past, but by the computational costs for searching the index of bubbles.

8.6 Conclusions for the discounted genetic model

The dynamic programming method allows determination of the optimal economic use of genetic information and includes the decision as to whether and when the value of the information justifies its cost.

While the optimal genetic progress possible is not significantly greater than the progress possible without the identification of genes, save for very short breeding programs, it is found that the present discounted value of optimal progress is significant, and justifies the payment of large royalties to find the underlying information.

8.7 Future research

The work provides a wide variety of opportunities for future research. The dynamic programming method itself is amenable to optimization, particularly with regard to search pattern and indexing methods. Additionally, it could be modified to take numerical derivatives allowing longer steps than the state-by-state search currently used, which could conceivably improve performance by orders of magnitude.

Many of the large number of identified genes in commercial livestock may be optimally utilized with this method. Multiple genes could be simultaneously selected, rather than the single gene considered here. Rather than the bounds used herein, Bayesian methods could reduce the number of animals which must be tested.

Finally, the discounted breeding problem is mathematically the same as the optimal growth problem of macroeconomics and the method could be used to solve more complicated growth models not amenable to analytic solution, as well as problems from game theory with repeated play and continuous choice spaces.

APPENDIX A GLOSSARY**allele**

The particular variant of the gene found at the QTL, such as *b* and *B*.

breeding value

The average effect on the trait in question of all genes that a parent passes on to offspring..

CDF

Cumulative distribution function.

diploid

Having two chromosomes. All animals, and many plants, are diploid.

dominance

A gene effect in which the first copy of the gene has a greater effect on the trait than the second. Dominance of zero means that the first and second copies are of equal value, full dominance indicates that the second gene is of no additional value, and overdominance means that the second gene is actually harmful.

ESR

The Estrogen Receptor Gene. Among other effects, the presence of this gene leads to larger litters for swine.

fixed cost

A cost that cannot be avoided in an enterprise, and does not vary.

genetic phase disequilibrium

When different truncation points are used for the genotypes, the result is a different truncation point for the polygenes in each group. In the next generation, the polygenes will have different means and variances in the different group. This creates a negative correlation between the major and polygenes known as Gametic Phase Disequilibrium.

genotype

The actual genetic status of the organism for a given locus. For example, *Bb*.

genotypic selection

Genotypic selection considers both the phenotypic and genotypic values. A value of $I = g + h^2(P - g)$ is used, where P is the phenotypic value, g the genotypic value, and h^2 the heritability.

heterozygote

In a diploid organism, a heterozygote for a given locus has a two different alleles at the locus in question.

heritability

The fraction of phenotypic variation in a trait that is due to genetics.

homozygote

An organism with two of the same allele at the locus in question.

intensity

A measure of how strongly the major gene's contributions considered.

locus

A point on a chromosome where a gene is located.

major gene

A gene with a large effect. It is assumed that the major gene can be identified, by QTL or other methods.

mass selection

Also phenotypic selection. Organisms are selected to reproduce based solely upon their own phenotypic value for the trait.

PDF

Probability density function.

phenotype

The observable trait. For example, the weight of an animal.

polygenes

Polygenes cannot be identified, but are seen only by their combined effect on phenotype. It is assumed that each of the polygenes has a small effect compared to the whole and to the major gene.

Quantitative Trait

A quantitative trait is one which takes a quantitative rather than qualitative value. Mendel's peas were qualitatively either wrinkled or not; they were one of the two types. A quantitative trait would instead be a measure of the height of the peas: a value in the range of three to six inches, for example, with the intermediate values being possible.

QTL

Quantitative trait locus, the locus that controls or affects a quantitative trait.

truncation point

A cutoff point on the selection criterion for selection. All creatures above this point breed, and those below do not.

truncation selection

Selection by accepting all animals above the truncation point for breeding, and none below.

variable cost

A cost that changes depending upon the decisions made.

APPENDIX B VARIABLE NAMES AND DEFINITIONS

The following variables have the following meaning unless otherwise specified:d

Variable	Meaning
a	the value of each allele of the major gene
A	polygenic breeding value
\hat{A}	estimated polygenic breeding value
b	the inferior allele
b_{mt}	weight used for genotype m selected in generation t
B	the breeding value. Also used to indicate the superior allele.
bestVal()	function returning the best possible value that could be reached for a state, as well as the choices made from that state forward
bubbletteAt()	function that finds the bubble and bubblette corresponding to the argument, creating the bubble if necessary
C_t	the choice space at time t
c	cost, typically of testing. Also used for choice
d	dominance effect
d_t	disequilibrium at time t
Δ	the granularity separating states
E	Environmental and random effects. Also the expectation operator
EOSHIFT()	Fortran intrinsic routine to shift matrix contents
ϵ_t	Lagrangian multiplier used for population constraint
F	fixed cost; also the cumulative distribution function; also an alternate objective function
f_{mt}	the fraction of the population in generation t of type m
ϕ	the probability density function of the standard normal distribution

Φ	The cumulative distribution function of the standard normal distribution
G	breeding value
g	genotypic value for the identified major gene
γ_t	Lagrangian multiplier used for change in \bar{A}_t
H_t	the Hamiltonian
h^2	heritability of the trait
I	breeding intensity
L	Lagrangian for optimal control treatment
λ	Lagrangian multiplier used for change in p_t
m	an indicator for genotype
∇	nabla, the Laplacian operator. Returns the gradient, or vector derivative, of a function with respect to the variables in question
μ	the mean of the distribution
p^c	the critical value for gene frequency above which testing cannot be profitable
p_t	the frequency of the major gene in generation t
P	the phenotypic value
π	profit
Q	the fraction of animals to be bred to produce the next generation
R	revenue
r	the discount rate
ρ	discount factor. $\rho = 1/(1+r)$
S	the state of the system
σ	variance, typically of the polygenic distribution
T	the final period. Also used as a function name for testing cost.
t	time
totGenerations	total number of generations used during the current iteration of the model
V	generic objective function
W	the present value of making default choices forever.
x_{mt}	the truncation point for genotype m in generation t
z_{mt}	the height (density) of the standard normal at x_{mt}

APPENDIX C SELECTED BREEDING PROGRAMS

Output from several different breeding programs, both from the algorithm developed herein and dekkers program, follow. The variable names displayed are the same as within the body, save that "AB-s" in Dekkers' output refers to $\bar{A}_{B,t}^s$ with the variations for the dams and inferior alleles being self explanatory. The variable d_t in the algorithm's output has the same value as $\bar{A}_{B,t} - \bar{A}_{b,t}$ in Dekkers' output. Finally, save for Figure C.1, Dekkers' output is normalized such that G_0 is 0, and the value of G_0 from the algorithm must be subtracted from Dekkers' G_t to compare those results to those of the algorithm.

```

parskip0
Gen  p          G          Abar    f1      f2      f3
 1 0.05000 -0.224999994039536 0.00000 0.500295 0.327620 0.185734
 2 0.08406 0.207228153944016 0.41520 0.428348 0.292564 0.181085
 3 0.12777 0.644088208675385 0.83020 0.380323 0.268445 0.176079
 4 0.18062 1.085383772850037 1.24507 0.346372 0.250374 0.170677
 5 0.24178 1.530712127685547 1.65982 0.321152 0.235861 0.164811
 6 0.31006 1.979507923126221 2.07448 0.301616 0.223486 0.158368
 7 0.38402 2.431057214736938 2.48904 0.285942 0.212349 0.151197
 8 0.46200 2.884529113769531 2.90353 0.272983 0.201804 0.143082
 9 0.54213 3.338995933532715 3.31793 0.261977 0.191297 0.133716
10 0.62241 3.793446540832520 3.73224 0.252411 0.180264 0.122658
11 0.70074 4.246827125549316 4.14646 0.243907 0.167976 0.109237
12 0.77496 4.698039531707764 4.56056 0.236185 0.153314 0.092416
13 0.84291 5.145965099334717 4.97451 0.229023 0.134196 0.070581
14 0.90244 5.589468955993652 5.38825 0.222212 0.105829 0.041521
15 0.95144 6.027318954467773 5.80160 0.215555 0.052549 0.007065
16 0.98777 6.457773208618164 6.21389 0.000000 0.000000 0.000000

```

Figure C.1 Dekkers' fifteen generation results for one state variable

```

Grain: 0.000024414
Best fitness in iteration 10 is 6.4577858
Gen  G          p          Abar    f1      f2      f3
0-0.225000000000000 0.05000 0.00000 0.500242 0.681628 0.148471
 1 0.207228082507341 0.08406 0.41520 0.428348 0.624505 0.120162
 2 0.644088447762011 0.12776 0.83021 0.380334 0.592696 0.081093
 3 1.085383905090603 0.18062 1.24508 0.346388 0.577214 0.026590
 4 1.530714849018840 0.24177 1.65983 0.321164 0.574245 0.000000
 5 1.979512353203580 0.31006 2.07448 0.301628 0.582620 0.000000
 6 2.431065606860210 0.38403 2.48905 0.285941 0.603053 0.000000
 7 2.884538601883760 0.46201 2.90353 0.272977 0.638110 0.000000
 8 3.339003137831710 0.54214 3.31793 0.261973 0.692868 0.000000
 9 3.793453953873290 0.62241 3.73225 0.252406 0.776677 0.000000
10 4.246831473037630 0.70073 4.14647 0.243906 0.907153 0.000000
11 4.698042689383100 0.77495 4.56057 0.236183 1.000000 0.000000
12 5.145969079352711 0.84290 4.97452 0.229028 1.000000 0.000000
13 5.589476982308782 0.90244 5.38826 0.222214 1.000000 0.000000
14 6.027330196175890 0.95144 5.80161 0.215550 1.000000 0.000000
15 6.457785764324584 0.98777 6.21390 0.000000 0.000000 0.000000
Best Steps are
3443 5233 7398 9903 12700 15730 18924 22206 25494 28702 31742 34525 36964 38971 40459
Gainof 0.00000004996758
used 55 bubbles

```

Figure C.2 The algorithm's fifteen generation results for one state variable without discounting

```

Grain: 0.012500000
5.000000000000000E-002 0.000000000000000 0.000000000000000 0.000000000000000
Best fitness in iteration 1 is 22.0902263
Gen  G          p          PrVal      Abar      f1      f2      f3
0-0.225000000000000 0.05000  22.18339  0.00000  0.810151 0.746834 0.140748
1 0.210880400647841 0.12500  20.74581  0.39838  0.645396 0.502086 0.104600
2 0.665703620980871 0.25000  19.29686  0.79070  0.471601 0.376133 0.052400
3 1.136052927601020 0.40000  17.75458  1.18605  0.381509 0.318578 0.000000
4 1.613684589765970 0.56250  16.13414  1.58243  0.305808 0.307771 0.000000
5 2.085352612453610 0.70000  14.41340  1.98535  0.268111 0.350599 0.000000
6 2.547784419502014 0.81250  12.63872  2.39153  0.238247 0.477341 0.000000
7 2.996643387908770 0.88750  10.82985  2.80289  0.223050 0.769336 0.000000
8 3.435650957312640 0.93750  9.03369  3.21690  0.210780 1.000000 0.000000
9 3.865587116246480 0.96250  7.28084  3.63434  0.205221 1.000000 0.000000
10 4.290779460831090 0.97500  5.61503  4.05328  0.205156 1.000000 0.000000
11 4.715414105590331 0.98750  4.07660  4.47166  0.200000 1.000000 0.000000
12 5.135357012325640 0.98750  2.69861  4.89161  0.200000 1.000000 0.000000
13 5.55299919060950 0.98750  1.52779  5.31155  0.200000 1.000000 0.000000
14 5.975242825796260 0.98750  0.61060  5.73149  0.200000 1.000000 0.000000
15 6.395185732531570 0.98750  0.00000  6.15144  0.000000 0.000000 0.000000

```

Figure C.3 The algorithm's fifteen generation results for one state variable with discounting

```

Grain: 0.000024414
5.000000000000000E-002 0.000000000000000 0.000000000000000 0.000000000000000
Best fitness in iteration 10 is 61.8135735
Gen  G          p          PrVal      Abar      f1      f2      f3
0-0.225000000000000 0.05000  41.51211  0.00000  0.732054 0.910998 0.123684
1 0.210997709997639 0.11277  41.70580  0.40461  0.565813 0.748802 0.054583
2 0.660548273643315 0.20530  42.00742  0.80790  0.451603 0.669880 0.000000
3 1.121667404894771 0.32083  42.41333  1.21125  0.373659 0.644285 0.000000
4 1.589420161377710 0.44712  42.92982  1.61586  0.319594 0.664779 0.000000
5 2.057828114151910 0.57058  43.57563  2.02254  0.281860 0.736252 0.000000
6 2.521870293598501 0.68042  44.38205  2.43166  0.255635 0.873739 0.000000
7 2.978632980520620 0.77087  45.39062  2.84320  0.237583 1.000000 0.000000
8 3.427333404611830 0.84099  46.65022  3.25684  0.225268 1.000000 0.000000
9 3.868627597450170 0.89292  48.21481  3.67217  0.216933 1.000000 0.000000
10 4.303831140035680 0.93013  50.14243  4.08877  0.211347 1.000000 0.000000
11 4.734379919192000 0.95623  52.49536  4.50627  0.207594 1.000000 0.000000
12 5.161500238088692 0.97424  55.34081  4.92438  0.205079 1.000000 0.000000
13 5.586139457391170 0.98655  58.75208  5.34287  0.203338 1.000000 0.000000
14 6.008900469620680 0.99475  62.80976  5.76152  0.202106 1.000000 0.000000
15 6.429727856944810 0.99998  67.60327  6.17974  0.000000 0.000000 0.000000
Best Steps are
4619 8409 13141 18314 23371 27870 31575 34447 36574
38098 39167 39905 40409 40745 40959
Gain of 0.208442715681670
used 44 bubbles

```

Figure C.4 The algorithm's fifteen generation results for one state variable with an infinite horizon

```

Grain: 0.000024414
5.0000000000000000E-002 0.0000000000000000 0.0000000000000000 0.0000000000000000
Best fitness in iteration 10 is 62.5053882
Gen G p PrVal Abar f1 f2 f3
0-0.2250000000000000 0.05000 38.33760 0.00000 0.769096 0.955718 0.118874
1 0.211071898260419 0.11829 38.38513 0.40193 0.591118 0.772122 0.039455
2 0.662835596831901 0.22197 38.52733 0.80185 0.466976 0.686045 0.000000
3 1.128209273144270 0.35369 38.75464 1.20137 0.382517 0.663755 0.000000
4 1.601277581167480 0.49893 39.06863 1.60181 0.324390 0.702924 0.000000
5 2.074667057843020 0.64119 39.48524 2.00407 0.284203 0.826228 0.000000
6 2.542076586173710 0.76731 40.03498 2.40842 0.256366 1.000000 0.000000
7 2.999550015885251 0.87002 40.76041 2.81454 0.236865 1.000000 0.000000
8 3.444997017973820 0.94719 41.71424 3.22140 0.222911 1.000000 0.000000
9 3.872923985374291 0.99998 42.98102 3.62294 0.200003 0.137502 0.089861
10 3.788940474319701 0.99998 46.88177 3.53895 0.200002 0.137501 0.089861
11 3.704955378451760 0.99999 51.06637 3.45496 0.200001 0.137501 0.089860
12 3.620969193000320 0.99999 55.55467 3.37097 0.200001 0.137501 0.089860
13 3.536982258449060 0.99999 60.36786 3.28698 0.200001 0.137500 0.089860
14 3.452994808886610 1.00000 65.52850 3.20300 0.200000 0.137500 0.089860
15 3.369007005251730 1.00000 71.06062 3.11901 0.000000 0.000000 0.000000
Switch to mass selection occurs at generation 9
Best Steps are
4845 9092 14487 20436 26263 31429 35636 38797 40959
Gainof 0.343696902227329

```

Figure C.5 The algorithm's fifteen generation results for one state variable with test costs

```

OPTIMAL SELECTION
Gen ps pd G AB-s Ab-s AB-d Ab-d A-bar Mu-BB Mu-Bb Mu-bB Mu-bb
0 0.0500 0.0500 0.00000 0.000 0.000 0.000 0.000 0.0000 0.250 0.000 0.000 -0.250
1 0.1123 0.1123 0.43583 0.132 0.211 0.132 0.211 0.4047 0.514 0.343 0.343 0.173
2 0.1776 0.1776 0.86908 0.336 0.417 0.336 0.417 0.8053 0.921 0.753 0.753 0.584
3 0.2514 0.2514 1.30555 0.548 0.621 0.548 0.621 1.2048 1.346 1.169 1.169 0.991
4 0.3388 0.3388 1.74698 0.759 0.823 0.759 0.823 1.6026 1.767 1.582 1.582 1.396
5 0.4406 0.4406 2.19328 0.966 1.025 0.966 1.025 1.9980 2.182 1.991 1.991 1.800
6 0.5572 0.5572 2.64332 1.171 1.225 1.171 1.225 2.3897 2.591 2.396 2.396 2.201
7 0.6899 0.6899 3.09555 1.372 1.424 1.372 1.424 2.7756 2.993 2.796 2.796 2.598
8 0.8543 0.8543 3.55040 1.560 1.659 1.560 1.659 3.1483 3.369 3.219 3.219 3.069
used 33 bubbles

```

Figure C.6 Dekkers' eight generation two dimensional results.

```

Best fitness in iteration 10 is 3.3252275
Gen G p diseq Abar f1 f2 f3
0-0.2250000000000000 0.05000 0.000000 0.00000 0.560408 0.439579 0.173783
1 0.210796833960690 0.11141 -0.078554 0.40509 0.333719 0.312250 0.169752
2 0.644005203295331 0.17526 -0.080710 0.80637 0.292406 0.279240 0.162149
3 1.080231894039040 0.24673 -0.071689 1.20687 0.287917 0.262912 0.149356
4 1.521547107384570 0.33195 -0.064454 1.60557 0.278012 0.249712 0.131337
5 1.967298638930641 0.43005 -0.058554 2.00228 0.273826 0.235871 0.103839
6 2.416956433397012 0.54227 -0.054582 2.39582 0.275888 0.213484 0.061538
7 2.869385559632381 0.67059 -0.053561 2.78409 0.312714 0.126095 0.033799
8 3.325227491620394 0.84239 -0.099433 3.15403 0.000000 0.000000 0.000000
Best Steps are
3927 6178 8697 11701 15159 19115 23638 29694
-2769 -2845 -2527 -2272 -2064 -1924 -1888 -3505
Gainof 0.000000360401403
used 119 bubbles

```

Figure C.7 The algorithm's eight generation two dimensional results

```

Grain: 0.000007092 0.000007092
Best fitness in iteration 12 is 4.1988626
Gen  G          p          diseq      Abar      f1      f2      f3
0-0.2250000000000000 0.050000 0.000000 0.000000 0.507241 0.393941 0.178734
1 0.209926298210333 0.099900 -0.066986 0.40998 0.319906 0.292885 0.177904
2 0.642068298367023 0.147650 -0.069632 0.81824 0.286508 0.267599 0.173985
3 1.076093866379310 0.199610 -0.063071 1.22629 0.277289 0.255240 0.167640
4 1.513348399949510 0.259140 -0.056993 1.63378 0.271415 0.245402 0.159502
5 1.953953905333980 0.326700 -0.052440 2.04060 0.265996 0.236248 0.149286
6 2.397586615871551 0.401780 -0.049256 2.44669 0.261466 0.227086 0.135889
7 2.843805193462730 0.483950 -0.047405 2.85183 0.256428 0.217328 0.117874
8 3.291678516606351 0.571660 -0.046525 3.25585 0.254797 0.202910 0.094630
9 3.740733377874474 0.664770 -0.048114 3.65835 0.317775 0.125389 0.032790
10 4.198862558203410 0.841860 -0.098604 4.02793 0.000000 0.000000 0.000000
Best Steps are
      14086    20818    28145    36538    46064    56651    68236    80604    93731    118701
      -9445    -9818    -8893    -8036    -7394    -6945    -6684    -6560    -6784    -13903
Gainof 0.000000862160000
used   1000 bubbles

```

Figure C.8 The algorithm's ten generation two dimensional results

```

OPTIMAL SELECTION
Gen ps  pd      G      AB-s  Ab-s  AB-d  Ab-d  A-bar  Mu-BB  Mu-Bb  Mu-bB  Mu-bb
0 0.0500 0.0500 0.00000 0.000 0.000 0.000 0.000 0.0000 0.250 0.000 0.000 -0.250
1 0.2077 0.1039 0.58288 0.239 0.329 0.160 0.226 0.5300 0.648 0.465 0.489 0.306
2 0.3226 0.2161 1.16300 0.520 0.596 0.429 0.497 1.0533 1.199 1.016 1.025 0.843
3 0.4801 0.3525 1.75154 0.798 0.854 0.703 0.762 1.5684 1.751 1.560 1.557 1.366
4 0.6416 0.5621 2.35003 1.073 1.108 0.967 1.017 2.0741 2.289 2.090 2.075 1.875
5 0.8773 0.8103 2.96408 1.321 1.428 1.214 1.312 2.5672 2.786 2.633 2.643 2.490

```

Figure C.9 Dekker's five generation results for four state variables

```

Grain: .000288595 .000288595 .000288595 .000288595
Best fitness in iteration 8 is 2.7424660
Gen  G          ps          ds          pd          dd          Abar      f1      f2      f3      f4      f5      f6
0 -.2250000000000000 .050000 .000000 .050000 .000000 .000000 .243758 .146354 .020854 .298442 .257699 .156719
1 .3548393617487033 .226840 -.095236 .077792 -.03925 .52865 .084213 .071556 .017946 .214476 .228828 .142772
2 .9383821315621663 .335930 -.075612 .20721 -.06003 1.05260 .065405 .057460 .010886 .211591 .217513 .122510
3 1.5287148319930500 .485710 -.055988 .35180 -.05628 1.56934 .064124 .045101 .000397 .229871 .208083 .074187
4 2.1267658427762200 .666660 -.028571 .54573 -.05108 2.07367 .070474 .014908 .002053 .299480 .109813 .026997
5 2.7424659645277600 .883680 -.107646 .81759 -.09783 2.56715 .000000 .000000 .000000 .000000 .000000 .000000
Best Steps are
      786    1164    1683    2310    3062
      -330   -262   -194    -99    -373
      270    718    1219    1891   2833
      -136   -208   -195   -177   -339
Gainof .000128267595050
used   137 bubbles

```

Figure C.10 The algorithm's five generation results for four state variables

```

Grain: .000144298 .000144298 .000144298 .000144298
Best fitness in iteration 9 is 2.7528003
Gen  G          ps          ds          pd          dd          Abar      f1      f2      f3      f4      f5      f6
0 -.2250000000000000 .050000 .000000 .050000 .000000 .000000 .243027 .146494 .020841 .570878 .242854 .157527
1 .3547001093435740 .226980 -.095236 .07778 -.04329 .52851 .089937 .045861 .021096 .325039 .212177 .126286
2 .4313171204277074 .380410 -.092762 .32385 -.11749 .50525 .072018 .035640 .015918 .284581 .180872 .104774
3 .5019678562657361 .564660 -.090809 .50288 -.11624 .48508 .056170 .026892 .011623 .242529 .149402 .083823
4 .5904584751569420 .735600 -.089595 .68433 -.11610 .48548 .045300 .021085 .008860 .209133 .125272 .068293
5 .7033443137813200 .858390 -.089056 .82517 -.11677 .53245 .000000 .000000 .000000 .000000 .000000 .000000
Best Steps are
      1573
      -660
      539
      -300
Gainof .010334353992258
used   1000 bubbles

```

Figure C.11 Sample output in which the solution was flushed from the cache

Actual mass selection solution:

Gen	G	p	diseq	Abar	prval	f1	f2	f3
0	0.7425	0.50000	0.000000	0.00000	0.00000	0.015055	0.013678	0.011281
1	19.7896	0.53094	-0.097469	18.76863	0.00000	0.014888	0.013566	0.011221
2	38.7867	0.56039	-0.165964	37.50606	0.00000	0.014747	0.013466	0.011161
3	57.7432	0.58839	-0.213643	56.22039	0.00000	0.014627	0.013374	0.011101
4	76.6664	0.61500	-0.246383	74.91777	0.00000	0.014522	0.013291	0.011041
5	95.5619	0.64025	-0.268447	93.60296	0.00000	0.000000	0.000000	0.000000

Grain: 0.000039063 0.000039063
Best fitness in iteration 9 is 99.3146521
Gain of 0.178556487891171

Gen	G	p	diseq	Abar	prval	f1	f2	f3
0	0.7425	0.50000	0.000000	0.00000	0.00000	0.014897	0.013097	0.012601
1	19.7132	0.52406	-0.079570	18.75367	0.00000	0.015429	0.013171	0.011545
2	38.7372	0.55512	-0.159297	37.50267	0.00000	0.013267	0.013483	0.013516
3	57.5063	0.56609	-0.159727	56.17540	0.00000	0.038490	0.001739	0.001242
4	78.6409	0.77047	-1.248906	75.56057	0.00000	0.022611	0.000000	0.000007
5	101.4601	0.89035	-3.434023	97.58621	0.00000	0.000000	0.000000	0.000000

Best Steps are
13416 14211 14492 19724 22793
-2037 -4078 -4089 -31972 -87911
bubblesused 4 83 118 62 250

Figure C.12 The algorithm's five generation results for the ESR without discounting

Actual mass selection solution:

Gen	G	p	diseq	Abar	prval	f1	f2	f3
0	0.7425	0.50000	0.000000	0.00000	207.50904	0.015055	0.013678	0.011281
1	19.7896	0.53094	-0.097469	18.76863	150.84701	0.014888	0.013566	0.011221
2	38.7867	0.56039	-0.165964	37.50606	99.10726	0.014747	0.013466	0.011161
3	57.7432	0.58839	-0.213643	56.22039	54.64985	0.014627	0.013374	0.011101
4	76.6664	0.61500	-0.246383	74.91777	20.40908	0.014522	0.013291	0.011041
5	95.5619	0.64025	-0.268447	93.60296	0.00000	0.000000	0.000000	0.000000

Grain: 0.000078125 0.000078125
Best fitness in iteration 8 is 210.6651033
Gain of 0.010618119251262

Gen	G	p	diseq	Abar	prval	f1	f2	f3
0	0.7425	0.50000	0.000000	0.00000	215.18129	0.018007	0.012252	0.011180
1	19.9536	0.54516	-0.147578	18.80569	159.27280	0.017396	0.011573	0.012148
2	39.0441	0.58445	-0.244844	37.55449	108.42993	0.015692	0.011784	0.013544
3	57.9455	0.61000	-0.274062	56.23906	65.14989	0.035967	0.000059	0.000077
4	80.5304	0.81500	-2.188984	77.11968	26.01465	0.020208	0.000000	0.000001
5	103.6136	0.91000	-4.564453	99.61944	0.00000	0.000000	0.000000	0.000000

Best Steps are
6978 7481 7808 10432 11648
-1889 -3134 -3508 -28019 -58425
bubblesused 2 1 0 2 31

Figure C.13 The algorithm's five generation results for the ESR with discounting

```

Actual mass selection solution:
Gen  G      p      diseq      Abar      prval      f1      f2      f3
0    0.7425  0.50000  0.000000  0.00000  554.07094  0.015055  0.013678  0.011281
1    19.7896  0.53094  -0.097469  18.76863  472.19984  0.014888  0.013566  0.011221
2    38.7867  0.56039  -0.165964  37.50606  379.16484  0.014747  0.013466  0.011161
3    57.7432  0.58839  -0.213643  56.22039  273.39358  0.014627  0.013374  0.011101
4    76.6664  0.61500  -0.246383  74.91777  153.11438  0.014522  0.013291  0.011041
5    95.5619  0.64025  -0.268447  93.60296  16.32479  0.000000  0.000000  0.000000
Best fitness in iteration 4 is 555.8123138
Gain of 0.134687684422261
Gen  G      p      diseq      Abar      prval      f1      f2      f3
0    0.7425  0.50000  0.000000  0.00000  586.21038  0.020452  0.013146  0.006948
1    20.2666  0.57625  -0.225000  18.84102  507.71405  0.016361  0.013227  0.008523
2    39.3808  0.62250  -0.308750  37.56780  419.07823  0.022853  0.009369  0.001150
3    59.0221  0.73375  -0.575000  56.29300  317.45287  0.019855  0.005153  0.010153
4    78.3745  0.81000  -0.827500  75.08830  200.82823  0.020458  0.000000  0.000002
5    101.9947  0.91000  -3.327500  98.00047  33.28498  0.000000  0.000000  0.000000
Best Steps are
      461      498      587      648      728
     -180     -247     -460     -662     -2662
bubblesused      1      0      0      31      250

```

Figure C.14 The algorithm's five generation results for the ESR with an infinite horizon

```

parskip0
Actual mass selection solution:
Gen  G      p      diseq      Abar      prval      f1      f2      f3
0    0.7425  0.50000  0.000000  0.00000  546.55699  0.015055  0.013678  0.011281
1    19.7896  0.53094  -0.097469  18.76863  466.05792  0.014888  0.013566  0.011221
2    38.7867  0.56039  -0.165964  37.50606  374.45456  0.014747  0.013466  0.011161
3    57.7432  0.58839  -0.213643  56.22039  270.16612  0.014627  0.013374  0.011101
4    76.6664  0.61500  -0.246383  74.91777  151.40872  0.014522  0.013291  0.011041
5    95.5619  0.64025  -0.268447  93.60296  16.16316  0.000000  0.000000  0.000000
Grain: 0.001250000 0.001250000
Best fitness in iteration 4 is 548.2789083
Gain of 0.132701402541670
Gen  G      p      diseq      Abar      prval      f1      f2      f3
0    0.7425  0.50000  0.000000  0.00000  578.20761  0.020452  0.013146  0.006948
1    20.2666  0.57625  -0.225000  18.84102  501.07174  0.016361  0.013227  0.008523
2    39.3808  0.62250  -0.308750  37.56780  413.85278  0.022853  0.009369  0.001150
3    59.0221  0.73375  -0.575000  56.29300  313.70202  0.019855  0.005153  0.010153
4    78.3745  0.81000  -0.827500  75.08830  198.60344  0.020458  0.000000  0.000002
5    101.9947  0.91000  -3.327500  98.00047  32.95543  0.000000  0.000000  0.000000
Best Steps are
      461      498      587      648      728
     -180     -247     -460     -662     -2662
bubblesused      1      0      0      31      250

```

Figure C.15 The algorithm's five generation results for the ESR with a differential interest rate so as to estimate elasticity

BIBLIOGRAPHY

- [Angel] Edward Angel and Richard E. Bellman, *Dynamic Programming and Partial Differential Equations*, Academic Press, New York, 1972.
- [Bellman] Richard E. Bellman and Stuart E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, 1962.
- [Boudarel] R. Boudarel, J. Delmas, and P. Guichet, *Dynamic Programming and its Application to Optimal Control*, Academic Press, New York, 1971.
- [Bertsekas] Dimitri P. Bertsekas, *Dynamic Programming and Stochastic Control*, Academic Press, New York, 1976.
- [Bulmer] M. G. Bulmer, *The Mathematical Theory of Quantitative Genetics*, Clarendon Press, Oxford, 1980.
- [Bulmer 71] M. G. Bulmer, The effect of selection on genetic variability, *American Naturalist*, 105:201-211, 1971.
- [Cassela] George Cassela and Roger L. Berger, *Statistical Inference*, Duxbury Press, Belmont, 1990.
- [Chanane] B. Chanane, Bilinear quadratic optimal control: a recursive approach, *Optimal Control Appl. Methods*, 18:273-282, 1997.
- [Chiang] Alpha C. Chiang, *Fundamental Methods of Mathematical Economics. Third Edition*, McGraw-Hill, New York, 1984.
- [Davidson] James Davidson, *Stochastic Limit Theory*, Oxford University Press, Oxford, 1994.
- [Dekkers 92] J. C. M. Dekkers, Asymptotic response to selection on best linear unbiased predictors of breeding values, *Animal Production*, 54:351-360, 1992.

- [Dekkers 98] J. C. M. Dekkers and J. A. M. van Arendok, Optimizing selection for quantitative traits with information on an identified locus in outbred populations. *Genetic Research, Cambridge*, 71:257-275, 1998.
- [Eikelenbroom] G. Eikelenbroom & D. Minkema, Prediction of pale, soft, exudative muscle with a non-lethal test for the halothane-induced porcine malignant hyperthermia syndrome, *Tijdschrift voor Diergeneeskunde*, 99:421-426, 1974.
- [Falconer] D. S. Falconer & Trudy F. C. Mackay, *Introduction to Quantitative Genetics. Fourth Edition*, Longman, Essex, 1996.
- [Freund] John E. Freund, *Mathematical Statistics*, Prentice Hall, Englewood Cliffs, 1992.
- [Gibson] J. P. Gibson, Short-term gain at the expense of long-term response with selection of identified loci. *Proceedings of the 5th World Congress on Genetics Applied to Livestock Production*, Guelph, 21:201-204, 1994.
- [Guidelines] *Guidelines for Uniform Swine Improvement Programs*, National Swine Improvement Federation, revised May 1996.
- [Hawkins] Richard E. Hawkins, working papers. Modeling Geometrically Increasing Multimodality.
- [Kamien] Morton I. Kamien and Nancy L. Schwartz, *Dynamic Optimization: The Calculus of Variations and Optimal Control in Economics and Management*. Elsevier Science Publishers B.V., 1981.
- [Larson] Robert E. Larson and John L. Casti, *Principles of Dynamic Programming*. Marcel Dekker, New York, 1982.
- [Lawrence 1599] John D. Lawrence, Marvin Hayenga, and Glenn Grimes, The U.S. Pork Industry Structure: a 1997 Snapshot, ASL-R1599, Department of Economics, Iowa State University, 1998.
- [Lawrence 1600] John D. Lawrence, Marvin Hayenga, and Glenn Grimes, The U.S. Pork Industry Structure: a 1997 Snapshot, ASL-R1600, Department of Economics, Iowa State University, 1998.

- [Øksendal] Bernt Øksendal, *Stochastic Differential Equations: An Introduction with Applications, Fourth Edition*, Springer, New York, 1995.
- [Pouliot] Michael Rene Pouliot, *A recursive quadratic programming approach to optimal control problems*. Masters' Thesis, Iowa State University, 1980.
- [Press] William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1986.
- [Puterman] Martin L. Puterman, *Dynamic Programming and Its Applications*. Academic Press, New York, 1978.
- [Rothschild 99] Max F. Rothschild and Graham S. Plastow, Advances in pig genomics and industry applications, *AgBiotechNet*, Vol 1 February, ABN 007, 1999.
- [Rothschild 94] M. F. Rothschild, C. Jacobson, D. A. Vaske, C. K. Tuggle, T. H. Short, S. Sasaki, G. R. Eckardt, and D. G. McLaren, A Major Gene for Litter Size in Pigs, *Proc. 5th World Congr. Genet., Appl. Livest. Prod.*, 21:115-228, 1994.
- [Short] T. H. Short, M. F. Rothschild, O.I. Southwood, D. G. McLaren, A. de Vries, H. van der Steen, G. R. Eckardt, C. K. Tuggle, J. Helm, D. A. Vaske, A. J. Mileham, and G. S. Plastow, Effect of the Estrogen Receptor Locus on Reproduction and Production Traits in Four Commercial Pig Lines. *Journal of Animal Science*, 75:3188-3142, 1997.
- [Varian] Hal Varian, *Microeconomic Analysis, Third Edition*, W. W. Norton & Company, New York, 1992.
- [Vertical Coordination] Vertical Coordination and Consumer Welfare: The Case of the Pork Industry. USDA AER #753, August, 1997.