Pacific Symposium on Biocomputing 15:43-53(2010)

OPTIMIZATION METHODS FOR SELECTING FOUNDER INDIVIDUALS FOR CAPTIVE BREEDING OR REINTRODUCTION OF ENDANGERED SPECIES

WEBB MILLER†
Center for Comparative Genomics and Bioinformatics, Penn State University Park, PA 16802

STEPHEN J. WRIGHT
Computer Sciences Department, University of Wisconsin Madison, WI 53706

YU ZHANG
Department of Statistics, Penn State University Park, PA 16802

STEPHAN C. SCHUSTER
Center for Comparative Genomics and Bioinformatics, Penn State University Park, PA 16802

VANESSA M. HAYES
Children's Cancer Institute Australia, University of New South Wales Randwick, NSW 2031, Australia

Methods from genetics and genomics can be employed to help save endangered species. One potential use is to provide a rational strategy for selecting a population of founders for a captive breeding program. The hope is to capture most of the available genetic diversity that remains in the wild population, to provide a safe haven where representatives of the species can be bred, and eventually to release the progeny back into the wild. However, the founders are often selected based on a random-sampling strategy whose validity is based on unrealistic assumptions. Here we outline an approach that starts by using cutting-edge genome sequencing and genotyping technologies to objectively assess the available genetic diversity. We show how combinatorial optimization methods can be applied to these data to guide the selection of the founder population. In particular, we develop a mixed-integer linear programming technique that identifies a set of animals whose genetic profile is as close as possible to specified abundances of alleles (i.e., genetic variants), subject to constraints on the number of founders and their genders and ages.

1. Introduction

It is generally agreed that techniques from genetics and genomics can be useful for understanding, and ideally sometimes preventing, the process of extinction[1,2,3]. One facet of species-conservation efforts, when appropriate, is to breed animals of an endangered species in captivity, e.g., in zoos and wildlife parks, with the goal of releasing them back into the wild at some time in the future. In some cases, the number of animals has dropped so low that all members of a species have been captured, as was done in North America for the California condor, red wolf and black-footed ferret. A better approach may be to identify species whose numbers are declining sharply but where more animals exist than can be supported in captivity, and to select from among the wild animals a representative subset to serve as the founder population for the captive breeding program.

In such cases, there is broad agreement among wildlife management officials that a goal should be for the founder population to capture at least, say, 95% of the genetic diversity of the wild population. However, there is disagreement over precisely what this means and how to achieve it. Perhaps the most common reasoning goes as follows[4]. Suppose a genomic position has two alleles in the population, with frequencies p and $1–p$. A founder population consisting of a random sample of n animals has $2n$ instances of that position, and the probability that all of them contain the first allele is p^{2n}. Thus, the probability that the n animals have at least one copy of each allele (rather than $2n$ copies of the same allele) is:

$$1 – p^{2n} – (1–p)^{2n}.$$

† Correspondence: webb@bx.psu.edu

If we want to obtain, with 95% certainty, both alleles at each random locus that occurs with a frequency of at least 0.05 in the wild population, then 30 founders are adequate, as is shown by evaluating the formula with $p = 0.05$ and $n = 30$. To have a 95% chance of capturing an allele with a frequency of only 1% requires 150 founders; 30 founders will probably not contain it.

The deficiency in this reasoning is that alleles don't occur at random in a wild population. Instead, a given allele is likely to occur more frequently in one sub-population than another, e.g., because of geographical barriers to gene flow. Thus, for example, picking all of the founders from one sub-population can easily miss alleles that are common overall. In practice, without an accurate survey of the genetic profile of the species, the number, degree of differentiation, and geographical locations of the relevant sub-populations may not be known.

Modern methods for sequencing and genotyping make it possible to directly assess the genetic diversity of a species and identify its population structure; sequencing a small sample of geographically distinct individuals can identify a number of the available genetic variants, and genotyping methods can then efficiently determine the genetic make-up of a large number of individuals population-wide. This paper describes how the resulting data can be used to select an optimal subset of the genotyped individuals that best matches specified allele frequencies, while satisfying additional constraints such as the total number of individuals selected, the number of males, and a suitable distribution of ages. Also, we show that allowing a bit of flexibility in the choice of founders can simplify the computation. In another formulation, we know the average genetic characteristics in sub-populations of the species, but not the genotypes of the individual candidates for the founder population.

One way to select the target genetic profile for the founder population is to match what is observed in living animals. In that case, random sampling would be justified if there is no population-genetic structure to the sampled population. Using the genotyping data required by our approach, one could run a program like STRUCTURE[5] to see if such structure exists.

The goal of recapitulating the distribution of alleles found in the overall population is by no means the only approach. We believe that in some cases a preferable goal may be to restore the balance that existed before the species was affected by the onslaught of industrial pollution, pesticides, disease, etc. in the last century or two. (We assume here that the pollution has been cleaned up, since we wouldn't want to undo any progress the species has made in dealing with it.) To support this approach, we are developing improved methods for sequencing the DNA from museum specimens[6,7], which can be used to determine past allele frequencies and population structure, and thereby identify a more natural target allele distribution for a founder population.

Another option for the target profile is where all alleles have frequency 0.5. For each locus, this minimizes the probability that an allele will be lost due to random genetic drift in the captive population. (This observation can be rigorously proved by martingale theory.) Moreover, this choice maximizes the genetic diversity in the captive population, since the probability that two randomly selected copies of a locus with minor allele frequency p are different is $2p(1–p)$, which is maximized when $p = 0.5$.

2. Methods

2.1. *The Data*

In our approach, several individuals of the species (perhaps even just a single individual) are sequenced using next-generation sequencing technology, until an adequate number of genomic differences has been identified. This step is not as easy as it sounds, because we assume that at the start of this process we know nothing about the genomic characteristics of the target species. In contrast, next-generation sequencing instruments and the available software for analyzing the data they produce are primarily designed for re-sequencing, i.e., where the goal is to look for small differences from an available "reference" genome sequence. Often, and also in our case, the main quest is for single-nucleotide polymorphisms, abbreviated SNPs. (Experts might cringe at this use of "SNP", instead requiring that a nucleotide difference be observed at a certain frequency, say $\geq 1\%$, before deserving the title of a polymorphism.) We call this initial stage for our approach "identifying SNPs without a reference". In more detail, the problem is to start with a specification for the desired number of SNPs, and automatically process short fragments of unannotated sequence data, identifying putative SNPs until enough have been found. We have developed software to solve this problem, but that is not the focus of this report. The outcome of this stage is a list of

genomic positions, each identified by, say, the 50 nucleotides on each side, where we predict that two distinct alleles are present in the population.

Once enough SNPs have been predicted, we design a custom genotyping array — a fabricated device that uses hybridization of DNA samples to the sequences flanking the putative sequence difference to determine which variant of each SNP is possessed by each assayed individual. For now, let us assume that each animal has two copies (maternal and paternal) of that genomic position; they can be the same (in which case we say the individual is homozygous for the SNP) or different (heterozygous). For each SNP, arbitrarily pick one of the two variants as the "reference allele", and suppose there are L variant nucleotide positions and that K individuals have been genotyped. The outcome of the genotyping step is an array, $\boldsymbol{A}$, with L rows and K columns, whose value A_{ij} at the intersection of row i and column j is the number of copies (0-2) of the i^{th} SNP's reference allele that was observed in the j^{th} individual. (Be warned that in other contexts it may be common for the roles of rows and columns to be reversed.)

There are many ways to formulate the problem of using genotyping data to select an optimal founder population for a captive-breeding program. We next describe a few of the possibilities.

2.2. *Formulation 1: Approximating Specified Allele Frequencies using Mixed-Integer-Programming Tools*

Suppose that from a pool of K genotyped individuals we want to select N (a specified size of the founder population) whose combined allele frequencies are as close as possible to a given ideal, e.g., the species' allele frequencies in the year 1900 as estimated from museum specimens. Let L denote the number of SNP positions, which we assume are biallelic (two variants), and for each SNP pick one allele arbitrarily for reference. Denote the target frequency of the reference allele for SNP i (where $1 \le i \le L$) by f_i with $0 \le f_i \le 1$. Thus, in the founder population of N animals, there ideally will be $b_i = 2Nf_i$ occurrences of that allele. Selecting a founder population is then equivalent to determining N binary variables $x_j \in \{0,1\}$ for $1 \le j \le K$, where $x_j = 1$ if the j^{th} animal is in the founder population and $x_j = 0$ otherwise. The requirement that there be N founders can be stated as $\sum x_j = N$. The number of occurrences of reference allele i in the founder population is $\sum \{A_{i,j} x_j : 1 \le j \le K\}$, which we want to be very close to the chosen target value b_i.

We will typically want to place additional constraints on the set of selected individuals. For instance, suppose we want to select exactly 50 individuals, of which 20 are males, and 10 are two years old. We form a 3-by-K array, $\boldsymbol{C}$, where first row is all 1s, the second row has a 1 in column j if the j^{th} genotyped individual is a male (0 for a female), and the third row has a 1 in positions corresponding to 2-year-olds (0 otherwise). Our constraints then have the form $\boldsymbol{Cx} = \boldsymbol{d}$, where $\boldsymbol{d} = (50, 20, 10)^{\text{T}}$.

Thus, our focus is on the following general formulation: We are given the $L \times K$ matrix $\boldsymbol{A}$ of allele frequencies $A_{i,j}$, the vector $\boldsymbol{b} = (b_1, b_2, \ldots, b_L)^{\text{T}}$ that indicates the desired total abundance of each reference allele in the founder population, and the $M \times K$ matrix $\boldsymbol{C}$ and vector $\boldsymbol{d} = (d_1, d_2, \ldots, d_M)^{\text{T}}$ that represent other constraints on the population (such as total population size, number of individuals of each age, number of each sex, and so on). We want to determine a vector $\boldsymbol{x} = (x_1, \ldots, x_K)^{\text{T}}$ of binary variables that indicate whether individual j $(j=1,2,\ldots,K)$ is included in the selected population, such that the desired allele abundances are matched as closely as possible subject to the specified constraints. Our problem formulation is thus

$$\text{Minimize}_{\boldsymbol{x}}\ \rho(\boldsymbol{Ax}-\boldsymbol{b}) \text{ subject to } \boldsymbol{Cx} \le \boldsymbol{d},\quad \boldsymbol{x} \in \{0,1\}^{\text{K}},$$

where $\rho(.)$ denotes a loss function that measures goodness of fit between two vectors of length L. (Although the constraints are written as inequalities, our discussion generalizes immediately to the case in which some or all are equality constraints.) The most useful goodness-of-fit measures ρ are the sum-of-squares function, the l_1 (sum-of-absolute-values) loss function, or the l_∞ (maximum residual) loss function, leading to the following three specific formulations:

$$\text{Minimize}_{\boldsymbol{x}}\ (1/2L)\Sigma_i (\boldsymbol{A}_{i\cdot}\,\boldsymbol{x}-b_i)^2 \text{ subject to } \boldsymbol{Cx} \le \boldsymbol{d},\quad \boldsymbol{x} \in \{0,1\}^{\text{K}},$$

(where $\boldsymbol{A}_{i\cdot}$ denotes the i^{th} row of $\boldsymbol{A}$);

$$\text{Minimize}_{\boldsymbol{x}}\ \Sigma_{i=1,2,\ldots,L}\ |\boldsymbol{A}_{i\cdot}\,\boldsymbol{x}-b_i|/L \text{ subject to } \boldsymbol{Cx} \le \boldsymbol{d},\quad \boldsymbol{x} \in \{0,1\}^{\text{K}}$$

and:

$$\text{Minimize}_{\boldsymbol{x}} \ \max_{i=1,2,\ldots L} |\boldsymbol{A}_{i.}\,\boldsymbol{x}-b_i| \text{ subject to } \boldsymbol{C}\boldsymbol{x} \le \boldsymbol{d}, \ \ \boldsymbol{x}\in\{0,1\}^{\mathrm{K}}$$

The objective in the least-squares formulation can be rewritten as $(1/2)\ \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ where $\boldsymbol{Q} = (1/L)\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ and $\boldsymbol{c} = -(1/L)\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{b}$, leading to a binary quadratic program. The l_1 and l_∞ formulations can be formulated as mixed-integer linear programs (MIP) by using some standard reformulation techniques. For the l_1 formulation, we introduce variables r_{i} and s_{i} to denote the positive and negative parts of $\boldsymbol{A}_{i.}\,\boldsymbol{x} - b_i$, respectively, and defining $\boldsymbol{r} = (r_1, r_2, \ldots, r_L)$ and $\boldsymbol{s} = (s_1, s_2, \ldots, s_L)$, we obtain the following reformulation:

$$\text{Minimize}_{\boldsymbol{x},\boldsymbol{r},\boldsymbol{s}} \ \boldsymbol{1}^{\mathrm{T}}(\boldsymbol{r}+\boldsymbol{s}) \ \text{ subject to } \ \boldsymbol{r}-\boldsymbol{s} = \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}, \ \boldsymbol{C}\boldsymbol{x} \le \boldsymbol{d}, \ \boldsymbol{r} \ge \boldsymbol{0}, \ \boldsymbol{s} \ge \boldsymbol{0}, \ \boldsymbol{x}\in\{0,1\}^{\mathrm{K}},$$

where $\boldsymbol{1}$ denotes the vector of length L whose elements are all 1. In fact, this is a binary MIP, since the solution contains a combination of continuous variables ($\boldsymbol{r}$ and $\boldsymbol{s}$) and binary variables ($\boldsymbol{x}$). For the l_∞ formulation, we obtain:

$$\text{Minimize}_{\boldsymbol{x},\eta} \ \eta \ \text{ subject to } -\eta\boldsymbol{1} \le \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b} \le \eta\boldsymbol{1}, \ \boldsymbol{C}\boldsymbol{x} \le \boldsymbol{d}, \ \boldsymbol{x}\in\{0,1\}^{\mathrm{K}},$$

where η is a scalar that captures the largest magnitude among elements in the vector $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$.

Commercial packages for solving MIP include CPLEX (www.ilog.com) and Xpress (www.dashoptimization.com), while CBC in the COIN-OR collection (https://projects.coin-or.org/Cbc) is a good open-source alternative. Many packages (including CPLEX and Xpress) now include support for integer quadratic programs as well as linear MIPs. Modeling languages are also available that interface to these solvers and allow users to specify their problem in an intuitive fashion. These include AMPL (www.ampl.com) and GAMS (www.gams.com). However, the problems in this paper have a simple enough form that modeling languages are not necessary unless one wishes to use the NEOS Solver, a web server for solving optimization problems at no cost (see www-neos.mcs.anl.gov).

MIPs are known to be extremely difficult to solve in general (they are NP-hard). However, for many practical problems, good solutions can be obtained in a reasonable amount of computing time. The algorithms underlying MIP codes are based on branch-and-bound strategies, cutting planes, and various other heuristics. When applied to the binary MIP above, branch-and-bound strategies construct a tree of relaxed problems, where in each node of the tree some of the binary variables x_j are fixed at 0 or 1 while the others are allowed to take on any value in the range [0,1]. This "relaxation" is a (continuous) linear program whose optimal value yields a lower bound on the optimal value of the original MIP. At each node of the tree, we form two child nodes by choosing one of the relaxed variables x_j and fixing it to 0 and 1, respectively. Note that the lower bound achieved at each child node must be at least as great as the lower bound at the current node. (At the root node of the tree, we relax all the variables and impose only the bounds $x_j \in [0,1]$, $j=1,2,\ldots,K$, along with the constraints $\boldsymbol{C}\boldsymbol{x} \le \boldsymbol{d}$.) The full tree would have 2^K nodes in total, but the hope is that we can avoid examining the vast majority of the tree by cutting off branches on which the lower bound is already greater than the value obtained at the "incumbent" – the feasible point with the best known objective value obtained to date.

Cutting planes are additional linear constraints added to the relaxed problem whose function is to exclude fractional solutions – those in which some components x_j are not either 0 or 1. Cuts can also be applied at lower nodes in the branch-and-bound tree. MIP software typically contains many heuristics for deciding on branching strategy (i.e. choosing the relaxed variable x_j on which to branch at each node), deciding what kinds of cutting planes to find and how often to look for them, and looking for candidate feasible solutions to use as incumbents. By setting options at input, users can control all these aspects of the codes.

The problems arising in this application have a property that makes them quite difficult to solve with standard MIP strategies. Because of the nature of the problem data, relaxation of the binary variables leads to a relaxed solution that has most of its components in the interior of the interval [0,1]. (This is true at most of the nodes of the branch-and-bound tree as well as at the root node.) Hence, the relaxed solutions are far from being feasible points for the original problem, and they produce only a weak lower bound on

the true objective. The upshot is that the branch-and-bound strategy is not able to exclude large parts of the tree from the search, and many nodes must be visited (and the relaxed problem at those nodes solved) before the lower bound increases to the point where a candidate solution can be declared to be nearly optimal. (This effect has also been observed in related contexts by D. Bienstock, in a personal communication.) It is likely that the heuristics for finding good candidate solutions in these codes do in fact generate near-optimal solutions after a fairly short time; the difficulty comes in verifying that indeed it is close to the best attainable. Generation of better lower bounds remains an open research question in computational mixed-integer linear programming.

When we use the mixed-integer quadratic programming formulation arising from the sum-of-squares loss function (see below), the same issue arises. However, in this case, in an unpublished manuscript dated 2009, D. Bienstock proposes to use lower bounds on the curvature of the quadratic objective to improve the quality of the lower bounds obtained at the nodes of the branch-and-bound tree. In our case, the matrix $\boldsymbol{A}$ (and $\boldsymbol{Q} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$) typically is well conditioned, so the bounds obtained by these means may be significantly stronger than the standard lower bound. We note that software frameworks for mixed-integer quadratic and nonlinear programs are much less prevalent than for linear MIPs, and this fact together with the large values of K and L for some data sets makes the logistics of developing a code for solving this problem quite daunting.

2.3. *Formulation 3: A Pure Integer-Programming Variation*

We have also explored simpler optimization problems related to selecting a set of individuals that satisfy requirements on their genotypes and other constraints. The hope is that the problem can be solved more efficiently than the problems discussed in the previous section, and that at least in some instances the solution will be adequate for the needs at hand.

For instance, formulating the computation as a linear programming problem whose only unknowns are the K binary variables x_{j} widens the domain of solvers that can be applied. One approach is as follows. Let $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{C}$ and $\boldsymbol{d}$ be as in Formulation 1. There, $\boldsymbol{A}$ and $\boldsymbol{b}$ have L rows and contribute to the objective function, while $\boldsymbol{C}$ and $\boldsymbol{d}$ have M rows and constitute the constraints. Let us assume that the constraints require that either $\boldsymbol{C}_{\mathrm{k}\cdot}\boldsymbol{x} = \boldsymbol{d}_{\mathrm{k}}$ or $\boldsymbol{C}_{\mathrm{k}\cdot}\boldsymbol{x} \geq \boldsymbol{d}_{\mathrm{k}}$ for $1 \leq k \leq M$ (where $\boldsymbol{C}_{\mathrm{k}\cdot}$ denotes the k^{th} row of $\boldsymbol{C}$). Form the $(2L+M-1) \times K$ array $\boldsymbol{A_2}$ and vector $\boldsymbol{b_2}$ as follows. For every row of $\boldsymbol{A}$, corresponding to the reference allele for a particular SNP, $\boldsymbol{A_2}$ contains that row plus a row for the other allele of that SNP. Thus the sum of the two rows has a 2 in every position. The corresponding two entries in $\boldsymbol{b_2}$ are the SNP's entry in $\boldsymbol{b}$ (giving the target abundance for the reference allele) and $2N$ minus that value. The other rows of $\boldsymbol{A_2}$ and $\boldsymbol{b_2}$ are taken from $\boldsymbol{C}$ and $\boldsymbol{d}$, omitting the constraint that the founder-population size is $\boldsymbol{N}$. The pure integer-programming approximation to Formulation 1 is:

$$\text{Minimize } \sum x_{\mathrm{j}} \text{ subject to } \boldsymbol{A_2 x} \geq \boldsymbol{b_2}, \quad \boldsymbol{x} \in \{0,1\}^{\mathrm{K}}$$

This reformulation makes the following changes. First, the size of the founder population is now variable. For instance, if each of 5 age groups is required to have 10 representatives among the founders, it might require 55 founders to also satisfy the other constraints. Also, where we formerly required the abundance of the reference allele for the i^{th} SNP to approximate b_{i} (and by inference the abundance of the SNP's other allele to approximate $2N-b_{\mathrm{i}}$), we now require the two alleles' abundances to equal or exceed those values. Finally, equality constraints in Formulation 1 are relaxed to inequalities. For instance, an original requirement of exactly 20 males now requires at least 20 males. Compared to the l_1 version of Formulation 1, this gives a problem with many fewer unknowns (but the same number of binary unknowns), though there are far more constraints. In this respect it is similar to the l_∞ version of Formulation 1.

An advantage of looking at the problem this way is that several approaches for cutting corners are suggested. First, it may be possible in practice to simply discard most of the constraints. For instance, if the only concern is to retain rare alleles, then constraints where the required allele frequency exceeds an appropriate lower bound could be removed. Another strategy is to replace the function being optimized (the number of founders) by a constraint that the founder population be "small enough". For instance, if we want to require that there be at least 10 individuals in each of five age groups (requiring 50 founders), we can add the constraint $\sum x_{\mathrm{j}} \leq 55$. Strategies like these can substantially expand the range of problem that are readily solved, as illustrated in Section 3.3, below.

2.4. *Formulation 4: Selecting Animals That Aren't Genotyped*

Another variant is to again fix the number of animals to be chosen, say N, and optimally select from the already genotyped pool *and/or ungenotyped wild individuals*. For ungenotyped animals, we assume that the genetic profiles of animals from different populations or geographic regions are already characterized from the genotyping. The point is not that individuals who are not genotyped are being selected. Rather, the issue is that representative individuals are being sampled from populations whose general properties are assumed to be known, even if the specific individuals are unknown.

Suppose we have L SNPs characterized for K populations (or geographic locations). Fix a "reference" allele for each SNP (say, the more-common allele), and let $p_{i,j}$ denote that allele's frequency for SNP i in population j, where $1 \le i \le L$ and $1 \le j \le K$. We assume that $p_{i,j}$ is known. If we want to sample x_j animals from population j, an optimal sampling strategy that yields the desired genetic diversity can be determined by the following criteria:

- The expected abundance of the reference allele for the i^{th} SNP closely approximates the target value b_i.
- If population j has c_j individuals, then $0 \le x_j \le c_j$.
- The total sample size is N.

Let $\boldsymbol{P} = \{p_{i,j}\}$ denote the $L \times K$ matrix of allele frequencies, $\boldsymbol{p}_i$ denote the i^{th} row of $\boldsymbol{P}$, and $\boldsymbol{b} = (b_1, b_2, \ldots, b_L)^T$. Also let $\boldsymbol{x} = (x_1, \cdots, x_K)^T$ denote the vector of unknown animal-counts from the various sub-populations. For SNP i, the expected count for the reference allele is $2\boldsymbol{p}_i\boldsymbol{x}$. Thus, the above criteria can be expressed as:

$$\text{Minimize}_{\boldsymbol{x}}\ \rho(\boldsymbol{Px}-\boldsymbol{b}) \text{ subject to } 0 \le x_j \le c_j \text{ and } x_j \text{ an integer for } 1 \le j \le K \text{ and } \sum x_j = N.$$

Here ρ can be any of the loss functions mentioned above. To include both selection (from genotyped animals) and sampling (from the wild), we can treat each genotyped animal as a "one-individual population", for which $p_{i,j}$ equals 0 (homozygote wild type), 0.5 (heterozygote) or 1 (homozygote mutation), and $c_j = 1$. In cases where the sub-population sizes c_j are fairly large, the restriction to integer solutions might not be so onerous as the constraint to binary unknowns in Formulation 2; rounding the entries of a solution to the continuous-variable problem might be good enough in practice. That is, the problem is solved by allowing each variable x_j to assume arbitrary real values between 0 and c_j, and then the optimal value replaced by the closest integer.

3. Experience

In essence, the genotyping data considered in this paper consists of a matrix with L rows and K columns, where each row corresponds to a genomic position where different nucleotides have been observed within a species, and each column corresponds to an individual animal of that species. Each entry is 0, 1, or 2, depending how many copies of the (arbitrarily chosen) reference allele for that difference are present in that individual. (For now, we are considering autosomal nucleotide polymorphisms, but in Section 3.4 we sketch what needs to be changed for other kinds of genomic polymorphisms, such as microsatellites.) Small numbers of these so-called SNPs (single nucleotide polymorphisms) have been used for various purposes related to wildlife management[8], but we anticipate the day, not long off, when next-generation sequencing and large-scale genotyping methods will be employed. In particular, our goal here is to use such data to select a set of animals with optimal genetic diversity that meets certain additional constraints, and we have described combinatorial optimization methods for several formulations. We now recount our experience to date with applying some of those methods to realistic (though mostly artificial) sets of data.

3.1. *Test Data*

Large genotype data sets will soon be available for endangered species, including the orangutan (personal communication from C. Bustamante and D. Locke). Currently, data for 384 SNPs and 322 individuals is available for a bird, the collard flycatcher[9]. Much larger data sets are available for some domesticated species. For instance, for cattle there is data for 37,470 SNPs and 497 individuals[10]. However, the most

extensive data sets are for humans, and we have employed these data to begin evaluating our proposed methods.

We used human genotypes from a preliminary data release from the 1000 Genomes Project[11,12], which we downloaded from ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/2009_04/**.** We identified over 4 million SNPs having data from each of the population groups CEU, JPTCHB, and YRI, which yielded complete data for 172 individuals. To experiment with various solution techniques we created data sets by randomly selecting L rows and K columns from the derived array of genotypes, for various values of L and for $K \leq 172$. In our tests, and we anticipate also in practice, L (the number of SNPs) is typically larger than K (the number of genotyped animals). When genders and ages of the individuals were needed, we randomly assigned genders and age groups 1 to 5 with equal probabilities. When a vector of target allele frequencies was needed, we scaled the observed frequencies in the selected genotypes to the desired size of the founder population. That is, for each row (SNP), we summed the numbers (0-2) in that row, multiplied by the size of the founder population, and divided by the number of columns (genotyped individuals).

3.2. *Results Obtained with Mixed Integer Programming Formulations*

For the problem of fitting a desired distribution of allele frequencies, we first wanted to see how well one can do using freely available software, without any attempt to tune the software's behavior to the particularities of our data, and making only modest use of computational resources. To do this, we attempted to solve instances of the l_1 version of Formulation 1 using a MIP solver called CBC run at the NEOS[13] server (without an "Options File" to adjust CBC's behavior). For each of several combinations of L and K, we requested a founder population with 20 males and 10 animals of each age (1 to 5), for a total of 50 founders. For 200 SNPs and 100 animals, the problem was solved to optimality in just over 3 minutes. When the number of SNPs was raised from 200 to 300, the time needed to find an optimal solution shot up to almost an hour. For 500 SNPs and 100 animals, or 300 SNPs and 172 animals, the computation exceeded our 5000-second time limit, producing a solution that is probably reasonable, but quite possibly not optimality.

We next report on results obtained with the commercial MIP solver CPLEX on the l_1 and l_∞ formulations, using a data set with 5000 SNPs and 172 individuals. Again, it was required to select 10 individuals of each age 1 through 5, of whom 20 would be males, for a total of six constraints. The objective was to make the frequency of alleles in the selected population as close as possible to the frequency in the overall population, in the l_1 or l_∞ sense. The modeling language GAMS[14] (www.gams.com) was used to define the model. (Direct use of GAMS models has the advantages that they allow variants of the models to be tried quickly, and that GAMS files can be submitted to the NEOS server.)

CPLEX, like other MIP solvers, allows many options and parameters to be set to non-default values, to improve their efficiency. Insight into the nature of problems being solved, and some trial-and-error and parameter tuning, can lead to dramatic improvements in run time over default values. For these problems, however, we able to obtain only modest improvements over default behavior by choosing alternative values of these parameters. Techniques that can handle the special characteristics of these problems (noted above) and produce verified optimal solutions in a reasonable amount of time are not available in current MIP solvers and are a topic of ongoing research. We believe, however, that the code is finding close-to-optimal solutions in reasonable time. As noted above, the difficulty comes only in verifying that they are indeed near-optimal, as we need to examine a large fraction of the nodes on the search tree to increase the lower bound to a level close to the best feasible solution obtained so far.

We mention for the record that the following non-default parameter values were used in the CPLEX runs reported here: cliques=–1 and covers=–1 (to turn off certain kinds of cut generation), dpriind=3 (to use steepest-edge pricing in slack space of the simplex method used at each node), varsel=4 (to force the use of pseudo-reduced costs as a criterion for branching), symmetry=5 (to generate symmetry-breaking cuts aggressively during the early stages of the solution process), mipemphasis=4 (to emphasize the search for better candidate solutions rather than improvement of the lower bound). We make no claims that this combination gives the best performance overall, or even that it is better than the default settings on these problems. CPLEX Version 11.2 was executed on both formulations for 15,000 CPU seconds (a limit prescribed by us) on a PC with an Intel Xeon quadcore CPU at 2.66 GHz, running Red Hat Enterprise Linux Server release 5.3, with 4GB of DDR2 memory at 800 MHz.

For the l_∞ formulation, the code found a point with objective value 9.9770, with a lower bound of 1.3901, within a few hundred seconds of CPU time. The candidate solution was found by running default

heuristics and adding cuts to the relaxed problem at the root node of the search tree. During the remainder of the 15,000 seconds of execution time, 1460 nodes of the search tree were examined and four progressively better candidate solutions were found. At termination, the best solution found had objective 9.4070, while the lower bound had increased only to 1.8060.

For the l_1 formulation, the code quickly identified a candidate solution with objective 2.2173, with a lower bound of .3931, again without performing any branching. During the remainder of the execution time, three more candidate solutions with progressively better objective values were identified, with the final solution having objective 2.2150, while the lower bound had increased to 0.4717. Only 201 nodes of the branch-and-bound search tree were evaluated.

We note that the solutions obtained with these two objectives were quite different. In fact, of the 50 individuals in each solution, only 19 were selected in both.

We conclude by reporting results obtained with the binary quadratic programming formulation. Note that the only variables in this formulation are the original 172 binary variables. (By forming the Hessian matrix $\boldsymbol{Q}$ and linear term $\boldsymbol{c}$ explicitly, we avoid the need to introduce any additional continuous variables.) We found that the results were improved slightly when we performed a simple transformation of $\boldsymbol{A}$ and $\boldsymbol{b}$, before forming $\boldsymbol{Q}$ and $\boldsymbol{c}$. Since a total of 50 animals are to be selected (that is, 50 of the components of $\boldsymbol{x}$ are 1 at the optimum, while the remainder are zero), we can subtract 1 from all elements of $\boldsymbol{A}$, while subtracting 50 from each element of $\boldsymbol{b}$. After this transformation, $\boldsymbol{A}$ contains elements –1, 0, and +1, and $\boldsymbol{Q}=(1/L)\boldsymbol{A}^T\boldsymbol{A}$ can have slightly better conditioning. We use this shifting procedure for the experiment reported below.

For this formulation, we wrote a C code to set up the problem and call CPLEX. Except for the termination criteria (which enforced a limit of 15,000 seconds on CPU time), default parameter settings for CPLEX's MIQP solver were used. Within a few seconds, the code finds an incumbent (the best feasible point encountered to date) with objective 3.9701. A total of 22 incumbents are found during the run. The last of these (which is the solution reported by the code) has an objective of 3.8113. After about a minute, the lower bound is approximately 1.71; it increases steadily but slowly thereafter to a final value of 1.8878. About 13,500,000 nodes of the branch-and-bound tree are examined during the run. CPLEX's mixed-integer QP solver makes much less use of cuts, and thus places much more reliance on branching as a means of solving the problem.

To compare the optimization methodology with a random strategy as a means for selecting a set of individuals that fits the target allele frequencies while satisfying the constraints, we programmed in MATLAB a code that generates 3414 feasible points for this test set at random. The mean of the objective values obtained from this process is 6.2729, with a standard deviation of 1.1798. The best objective found was 4.8353, which is significantly worse than the solution obtained by the optimization code after just a few seconds of run time.

3.3. *The Pure Integer-Programming Variation*

For a given set of genotyping data, finding a "sweet spot" for the many combinations of potential LP shortcuts, available solvers, and choices of solver options will require some effort. As might be expected of a computational problem with exponential time complexity, seemingly minor changes in the approach often mean the difference between a quick solution and failure of the computation to terminate in a reasonable time. However, we now report one simple experiment that hints at what can be achieved.

In Section 3.2 we noted failure of a straightforward attempt to run a NEOS MIP solver on random data for 300 SNPs and 172 animals. To experiment with reformulations of the problem, we generated random data for 2000 SNPs and 172 animals and applied the following shortcuts to the approach outlined in Section 2.3. First, we permitted up to 55 founders (while continuing to require at least 10 in each of the five age groups). Second, we retained constraints on the allele abundance only when the lower bound was 5 or less, leaving 146 of the original 4000 allele-abundance constraints. Using C-language programs that we wrote to convert genotyping data to MPS format, we submitted the data to the SCIP[15] optimizer at the NEOS server (with no "Parameter file"). In around 30 seconds, SCIP found a feasible solution containing 55 founders. We discovered that 16 of the original alleles whose constraints had been removed had less than 5 occurrences in the computed founder-set. All of those had at least 3 occurrences, which might be considered adequate under certain conditions. An alternative is to add the requirement that each of those 16 alleles occur at least 5 times to the 146 allele-abundance constraints, and re-solve the problem.

We also experimented with an even simpler integer-programming formulation, using a real-world data set. The objective was simply to guarantee a minimum number of occurrences of each allele, using genotyping data from a 96-SNP array that we designed for the Tasmanian devil (*Sarcophilus harrisii*), a species that is gravely threatened by extinction from a transmissible tumor[16]. We used the array to genotype 85 animals, and for the following analysis used the 69 SNPs that appeared in more than one animal. The problem we investigated was to select a set of animals that contains each allele at least twice, which corresponds to a binary linear programming problem of minimizing Σx_j subject to $\boldsymbol{Ax} \geq \boldsymbol{2}$, $\boldsymbol{x} \in \{0,1\}^K$, where $\boldsymbol{2}$ denotes the vector containing 2s. For our 69-by-85 array, $\boldsymbol{A}$, linear programming selected four animals. Interestingly, a simple greedy algorithm[17] that repeatedly picks the animal that adds the most "uncovered" alleles, selected six animals. With four randomly selected animals, an average of only 22.7 of the 69 alleles appeared at least twice. Looked at another way, random selection picked an average of 31.9 animals before reaching 2-fold representation of every allele. What makes the problem difficult for random sampling is that a few of the alleles appeared only a few times among the 85 animals. (Extensive comparisons of the greedy and random selection strategies for a very similar problem have been published by others[18].)

3.4. *Handling Other Types of Polymorphisms*

Our approach has been described under the assumption that each genetic marker has two possible states, and that each individual has two copies. Minor adjustments are required for other cases, such as microsatellites with multiple states, or sex-chromosome or mitochondrial markers where an individual may have fewer than two copies. For instance, for a microsatellite with three observed lengths, we could devote three rows of the genotypes array, one for each length. For each column (individual) the values in those rows will each be 0, 1, or 2, and the sum of the three values will be 2 (except perhaps for microsatellites on a sex chromosome). This works for both the linear-programming and the quadratic-programming formulations.

4. Conclusion

We have developed and evaluated several methods that can use high-throughput genotyping data to select a set of animals that best approximates a specified allele profile, subject to additional constraints; the selected animals might be used as founders for a captive breeding program or reintroduced into a former range for that species, for example. We anticipate that the costs of sequencing and genotyping will continue to plummet, making it feasible to gather genome-wide population-genetic data from hundreds of animals from each of many species. One can easily imagine having genotypes for many thousand SNPs in hundreds of individuals from an endangered species, and using that information to select founders. The number of SNPs required to reliably represent genome-wide variation is liable to depend heavily on the particular species (not to mention how one interprets "reliably represent"). For humans it has been estimated[19] that, among all of the 6 million common (minor-allele frequency ≥ 0.1) European SNPs in the human genome, about 80% could be ascertained at $\rho^2 > 0.8$ through pairwise linkage disequilibrium (LD) by genotyping 0.8 million European common and non-redundant variants in the database dbSNP. To cover a similar percentage of common SNPs in African-Americans, the number increases to 1.1 million SNPs. For a highly bottlenecked species with more extensive LD, the required number of SNPs could be much less.

In any case, even with the available technologies, affordable approaches are possible. For a few tens of thousands of dollars it is possible to sequence transcripts (cDNA) from several individuals, from which polymorphic amino-acid positions can be identified. If the number of those differences exceeds the desired array capacity, computational analysis could reduce them to the differences that show signs of being functionally important[20]. For an additional few tens of thousands of dollars, a custom genotyping array can be purchased and applied to genotype those differences on several hundred animals. Techniques described in this paper, such as the mixed-integer-programming technique, can then perform an optimal, or at least near-optimal, selection of founders, in the sense of capturing desired abundances of the putatively functional protein variants.

Departure of the target allele distribution from the distribution in the overall population increases the value of using an approach like ours, compared to simple random selection. For instance, we might want to avoid hybridized alleles (e.g., cattle genes in American bison). More generally, the belief that the goal of conservation efforts is to restore ecosystems to their "natural state", which to some writers means their state before any human intervention, has been widely discussed[21]. Several publications[22,23] have used museum

specimens to determine earlier genetic profiles. However, this has typically been done using only tiny amounts of hyper-variable sequence data, such as 500 bp from the mitochondrial control region or a small number of microsatellites. Another scenario with differing allele profiles might be importing animals to bring a struggling population up to a viable size when we want to retain its unique genetic make-up and/or reduce the threat of outbreeding depression.

Other decisions related to species conservation have been approached with computational methods. One such problem is selecting a set of species to be preserved. The problem has been formalized as optimally selecting a specified number of species, given a phylogenetic tree relating a more inclusive set of species and with branch lengths that measure inter-species differences. A natural objective is to maximize the sum of branch lengths in the induced subtree[24]. A simple greedy algorithm guarantees an optimal solution[25], and dynamic-programming algorithms solve several generalizations[26.27].

A set of individuals within a species is less well represented by a weighted tree, and other ideas have been applied to selecting intervals, with "maximum diversity" a frequent goal. For instance, given a set of genetic markers (SNPs, microsatellites, allozymes, etc.) one could try to maximize the number of observed alleles represented in the selected set of individuals[28]. This is essentially just the infamous Minimum Set Cover problem[29], which is NP-complete and hence solved in biodiversity practice[17] and studied in computer science theory[30] using approximation methods (including linear programming). The problem studied here, with the added complexity of an arbitrary target distribution and constraints on gender and age, is correspondingly more challenging.

A somewhat different class of optimization problems arises for management of a captive breeding program, where in addition to genotypes one has access to genealogies. As with selection of founders, there is tension between the potential goals of maximizing diversity and maintaining allele frequencies[31]. Combinatorial optimization methods have been proposed for designing breeding programs[32,33].

To keep pace with plummeting costs for sequencing and genotyping, more work is needed to extend the computational approaches described in this paper. Cutting-edge research in optimization algorithms that exploits the particular characteristics of this family of optimization problems may be needed to best utilize cutting-edge data-producing technologies.

Acknowledgments

We thank Belinda Giardine for preparing the 1000-Genomes data set. Oleg Shylo and the reviewers provided a number of helpful suggestions. W.M. and Y.Z. are supported by grant HG-002238 from the National Human Genome Research Institute. S.J.W. has support from NSF grants DMS-0427689, CCF-0430504, DMS-0914524, and DOE grant DE-FG02-04ER25627. S.C.S. is supported by the Gordon and Betty Moore Foundatation, and V.M.H. is a Cancer Institute of New South Wales Fellow.

References

1. R. Frankham, J. D. Ballou and D. A. Briscoe, *Introduction to Conservation Genetics,* Cambridge University Press (2002).
2. J. Höglund, *Evolutionary Conservation Genetics,* Oxford University Press (2009).
3. G. Bertorelle, M.W. Bruford, H. C. Hauffe, A. Rizzoli and C. Vernise, *Population Genetics for Animal Conservation*, Oxford University Press (2009).
4. D. R. Marshall and A. D. H. Brown, in *Crop Genetics Resources for Today and Tomorrow* (O. H. Frankel and J. G. Hawkes, eds.), Cambridge University Press, 53-80 (1975).
5. J. K. Pritchard, M. Stephens and P. Donnelly, *Genetics* **155**, 945-959 (2000).
6. W. Miller, D. I. Drautz, A. Ratan, et al., *Nature* **456**, 387-390 (2008).
7. W. Miller, D. I. Drautz, J. Janecka. et al., *Genome Research* **19**, 213-220 (2009).
8. J. Slate, J.Gratten, D. Beraldi, et al., *Genetica* **136**, 97-107 (2009).
9. N. Backström, N. Karaiskou, E. H. Leder et al., *Genetics* **179**, 1479-1495 (2008).
10. The Bovine HapMap Consortium, *Science* **324**, 528-532 (2009).
11. B. M. Kuehn, *JAMA* **300**, 2715 (2008).
12. N. Siva, *Nat. Biotech.* **26**, 256 (2008).
13. J. Czyzyk, M. Mesnier and J. J. Moré, *IEEE Comp. Sci. Eng.* **5**, 68-75 (1998).
14. A. Brooke, D. Kendrick and A. Meeraus, *GAMS: A User's Guide*, The Scientific Press (1988).

15. T. Achterberg, *Constraint Integer Programming*, Ph. D. dissertation, TU Berlin (2007). Available at http://scip.zib.de/documentation.shtml.
16. H. V. Siddle, A. Kreiss, M. D. Eldridge et al., *Proc. Natl. Acad. Sci. USA* **104**, 16221-16226 (2007).
17. B. Gouesnard, T. M. Bataillon, G. Decoux et al., *J. Hered.* **92**, 93-94 (2001).
18. H. I. McKhann, C. Camilleri, A. Bérard, et al., *Plant J.* **38**, 193-202 (2004).
19. C. S. Carlson, M. A. Eberly, M. J. Rieder et al., *Nat. Genet.* **33**, 518-521 (2003).
20. P. Ng, S. Levy, J. Huang et al., *PLoS Genetics* **4**, e1000160. doi:10.1371/journal.pgen.1000160 (2008).
21. P. I. Angermeier, *Conserv. Biol.***14**, 373-381 (2000).
22. H. C. Rosenbaum, M. G. Egan, P. J. Clapham et al., *Conserv. Biol.* **14**, 1837-1842 (2000).
23. A. Koblmüller, M. Nord, R. K. Wayne and J. A. Leonard, *Mol. Ecol.* **18**, 2313-2326 (2009).
24. D. P. Faith, *Biol. Conserv.* **61**, 1-10 (1992).
25. M. Steel, *Syst. Biol.* **54**, 527-529 (2005).
26. F. Pardi and N. Goldman, *Syst. Biol.* **56**, 431-444 (2007).
27. B. Q. Minh, F. Pardi, S. Klaere and A. von Haeseler, *IEEE/ACM Trans.Comp, Biol. Bioinf.* **6**, 22-29 (2009).
28. D. J. Schoen and A. H. D. Brown, *Proc. Natl. Acad. Sci. USA* **90**, 10623-10627 (1993).
29. R. Karp, *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, eds), Plenum Press, 85-103 (1972).
30. D. Hochbaum, *Approximation Algorithms for NP-Hard Problems.* PWS Publishing, Boston (1996).
31. M. Saura, A. Pérez-Figueroa, J. Fernándes et al., *Conserv. Biol.* **22**, 1277-1287 (2008).
32. J. Vales-Alonso, J. Fernándes, F. J. Gonzáles-Castaño and A. Caballero, *Math. Biosci* **183**, 161-173 (2003).
33. T. Y. Berger-Wolf, C. Moore and J. Saia, *J. Theor. Biol.* **244**, 433-439 (2007).