diversity

Technical Note

A Fast Method for the Selection of Samples in Populations with Available Genealogical Data

Dalibor Hršak [1], **Ivan Katanić** [2] **and Strahil Ristov** [1,*]

[1] Ruđer Bošković Institute, 10000 Zagreb, Croatia; dalibor.hrsak@irb.hr
[2] Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia; ivan.katanic@gmail.com
* Correspondence: ristov@irb.hr

Abstract: Optimal selection of samples in populations should provide the best coverage of sample variations for the available sampling resources. In populations with known genealogical connections, or pedigrees, this amounts to finding the set of samples with the largest sum of mutual distances in a genealogical tree. We present an optimal, and a faster sub-optimal, method for the selection of K samples from a population of N individuals. The optimal method works in time proportional to NK^2, and the sub-optimal in time proportional to NK, which is more practical for large populations. The sub-optimal algorithm can process pedigrees of millions of individuals in a matter of minutes. With the real-life pedigrees, the difference in the quality of the output of the two algorithms is negligible. We provide the Python3 source codes for the two methods.

Keywords: genealogical tree; sampling plan; optimal population coverage; pedigree sampling; mitochondrial DNA; Y chromosome

check for updates

Citation: Hršak, D.; Katanić, I.; Ristov, S. A Fast Method for the Selection of Samples in Populations with Available Genealogical Data. *Diversity* **2022**, *14*, 150. https://doi.org/10.3390/d14020150

Academic Editor: Michael Wink

Received: 27 January 2022
Accepted: 17 February 2022
Published: 20 February 2022

1. Introduction

The DNA sequencing of sample individuals is a standard method for obtaining information about genetic variability in a population [1,2]. As a rule, the number of samples that can be harvested is limited with the available resources, and the decision on a sampling strategy remains an important research subject [3]. A unique aspect of the sampling problem arises when the genealogical data are available for a population, that is, the relations between the individuals are organized in a pedigree. This is the usual case in commercial breeding and conservation contexts.

When the sequences of interest are mitochondrial DNA or Y chromosomes, that is, when analysing single-gender lineages, the genealogical information enables the selection of a particular set of individuals for sampling that would lead to the largest expected coverage of the genetic variability. Clearly, the set of individuals that is expected to cover the largest variability is the set of mutually most distant individuals in a pedigree. The task of selecting the optimal set of K individuals for sampling is therefore translated into an algorithmic problem of selecting the K most distant individuals in a (pedigree) tree of size N. This problem was addressed in Magellan software for the analysis of maternal lines in a pedigree [4], where a transparent and user-friendly method was implemented to solve the (quasi)optimal set selection problem on the small and medium-sized pedigrees. However, the algorithm implemented in the Ref. [4] has a time complexity proportional to N^3. As a result, it would take hours and days to process pedigrees with the number of individuals in the orders of magnitude of 10^5 and 10^6, respectively. Magellan software was successfully used in population genetics analysis [5], but the sample selection is possible only on the comparatively small pedigrees.

In the present work, we focus on improving the speed of execution in order to enable processing of even the largest pedigrees. We provide a code for a (slightly) sub-optimal algorithm for the selection of samples that can process millions of individuals in the matter of minutes. Mostly for the purposes of comparison, we also provide a code for the optimal algorithm. The optimal algorithm is considerably slower for larger K, while on the real data, the difference in the quality of the output between the sub-optimal and optimal algorithms is negligible.

In the rest of the paper we describe the two algorithms, give details of software usage, and present the results for simulated and real pedigree examples.

2. Methods

2.1. Software Availability

The Python3 source code of our software is freely available at https://github.com/lisp-rbi/pedigree-sampling (accessed on 26 January 2022) (for the interested reader, we have included the essential code in C^{++}, too).

2.2. Algorithms

The input parameters for the sampling selection problem are the pedigree file and the integer K—the intended number of individuals for the sampling (In our software, two methods of calculation are bundled together, and as a result, the chosen method must also be provided as the input parameter). We consider the pedigree as a forest of separate trees, where each tree represents one single-gender lineage that propagates from one founder dam or sire. Depending on the sizes of the distinct trees in the whole pedigree, a set of K values for each separate tree is calculated. The K value for i-th tree is denoted with K_i. If the size of the whole pedigree is denoted with N, then the size of the i-th tree is denoted with N_i. In actual implementation N_i and the proportional K_i values are calculated, and separate lineages are processed sequentially. However, for the sake of the simplicity of the exposition, in the following we shall treat a pedigree as a single tree and use N, K notation. Then the statement of the algorithmic problem is: *In a pedigree tree of size N, find the subset of K individuals such that no other subset of K individuals has a larger sum of mutual pairwise distances.*

We provide two algorithmic solutions, a sub-optimal greedy, and the optimal. Their principles of operation are explained as follows.

Greedy: The greedy algorithm builds the solution subset incrementally. In the first stage, the subset is initialized to the most distant pair of individuals. Finding the pair is a classical problem that can be solved in time proportional to N. The second stage has $K - 2$ steps. In each step we find an individual not yet in the subset with the largest cumulative distance to all the individuals in the subset. At the end of the second stage, the subset has reached size K and we are done.

Steps of the second stage can be performed efficiently if we maintain, for each individual, the cumulative distance to all the individuals in the subset. As the number of steps is proportional to K steps, the total time complexity of the second stage, and the algorithm, is proportional to NK.

Optimal: The optimal algorithm expresses the total distance to be maximized as a sum of values assigned to edges. Each edge is assigned a value corresponding to the number of pairs of individuals from the subset such that the individuals are on different sides of the edge. Note that the value assigned to an edge is simply the product of two numbers: counts of individuals from the subset for each of the two subtrees that the edge connects. The sum of those values over all the edges is equal to the total pairwise distance between individuals in the subset.

We observe the following: if we choose an edge and decide how many individuals from the subset will be on each side of the edge, the problem breaks into two. As the value of the chosen edge is finalized, we can remove it and continue by finding a fixed-size subset of individuals from each of two subtrees generated by the edge removal. The two subtree problems can be solved independently. This is true because for an edge of a subtree it holds that all the individuals outside the subtree are on the same side of the edge. Hence, to compute the contribution of a subtree, we do not need to know the exact choice of individuals outside the subtree, but only their count.

This problem is solved using dynamic programming. A subproblem in the dynamic programming algorithm will be a subtree and an integer $k \leq K$. The answer to a subproblem is the sum of values assigned to the edges in the subtree after optimal choice of k individuals. It is important to note that the choice of individuals from the rest of the tree does not matter. Only their count matters, which is equal to $K - k$.

By removing edges in a specific order (rooted, depth-first traversal) we can see that the number of different subproblems is proportional to NK. Each of them can be solved in time proportional to K as it involves removing an edge and considering all choices for the number of individuals on each side of the removed edge. The total time complexity of the algorithm is then proportional to NK^2.

The selected subsets of individuals are associated with the *score* values, calculated as the sum of all pairwise distances among the selected individuals. For the measured scores on real and simulated pedigrees, the difference between optimal and greedy method is observed to be very small, less than 0.01% in all cases. Incidentally, the differences are larger on trees with long sequences of descendants with only one individual per generation—an unlikely case in the real-life populations.

It should be noted that we use the term "optimal" in the context of the algorithmic solution and that this does not necessarily imply that the actual set of the chosen individuals will indeed be optimal for the largest coverage of the variability. We merely expect it to be, since the mutual distance maximisation is the natural approach to the problem of variability coverage.

2.3. Usage

2.3.1. Defining the Reference Population

The selection for sampling must be restricted to the accessible individuals, that is, the alive ones, or those with the stored tissue samples. Such individuals form the reference population, and they have to be separately annotated in a pedigree. For this purpose, we use an additional column in the pedigree file labelled *live*, where a non-empty *live* field indicates that the individual belongs to the reference population.

Therefore, the first step in using our software should be to define the reference population through assignment of the *live* values to individuals. If the *live* column is not found in the subsequent processing, the entire pedigree is treated as the reference population. The results obtained in this way may not have biological relevance.

The simplest way to designate the reference population in a pedigree is to assign the *live* status to all individuals born after a given year. We provide the script *setrefpop.py* that performs this function. If the reference population is otherwise defined, the user should employ a customized function.

Besides defining the reference population, the *live* attribute is used for the selection of gender lineages, as explained below.

2.3.2. Selecting the Gender Lineage

The main motive for the construction of our software was to support the analysis of mitochondrial DNA variability. As a result, the default usage is adjusted for the dam lineages. However, the algorithms are the same, regardless of the gender, and our software can be easily applied to the analysis of sire lineages and Y chromosome variability. In order to do that two actions are required.

Firstly, the reference population must include only sires. The *setrefpop.py* script includes this option and can assign *live* value only to male individuals, which can be combined with the referent year setting.

Secondly, the user must set the options for the preprocessing script to include only male individuals. The details on preprocessing are given in the following section.

2.3.3. Preprocessing and the Main Processing

Our algorithms work with a standard edge list graph representation of a pedigree tree. We provide the script *preprocess.py* that transforms the input pedigree file into the appropriate input for the main processing program. Along with the list of edges, the transformed file contains the explicit list of individuals that belong to the reference population. In order to process the paternal lineages in a pedigree, the user must input the `--parent father` option, while the `--parent mother` option is the default mode. This is clearly indicated in the instructions included with the software.

The final processing is performed with *main.py* script that requires a preprocessed input file, a value for K, and the choice between the greedy or optimal method.

3. Results and Discussion

In Table 1 we present the experimental results for the two simulated and one real large pedigrees. The simulated pedigrees were produced using the program for artificial pedigree construction that is included with our software. The parameters of pedigree simulation include the final pedigree size and the number of founders. The real pedigree is the large anonymized Czech Holstein cattle pedigree [6].

Table 1. Experimental results for the processed pedigrees. N denotes the size of the pedigree; N_F denotes the number of founder individuals, separately for the maternal and paternal lines; K and K^* denote the submitted and the actual number of selected individuals, respectively; t_G, t_O and t_P denote times measured for the greedy processing, optimal processing, and the preprocessing, respectively.

	N	***N_F*** **Dams/Sires**	***K***	***K****	***t_G***	***t_O***	***t_P***
Simulated	1 M	50/50	10^3	977	18 s	150 s	8 s
		"	10^4	9978	92 s	170 min	
		200/200	10^3	884	11 s	32 s	
		"	10^4	9896	28 s	614 s	
	20 M	200/200	10^3	884	290 s	742 s	170 s
		"	10^4	9902	643 s	206 min	
		500/500	10^3	558	228 s	461 s	
		"	10^4	9749	395 s	45 min	
Real	21.1 M	514,297/146,217	10^4	2	135 s	135 s	145 s
			10^5	1140	135 s	138 s	
			2×10^5	13,150	140 s	164 s	
			3×10^5	43,930	143 s	183 s	
			10^6	498,570	175 s	438 s	
			2×10^6	1,291,050	209 s	1020 s	

We measured the processing speed for the two algorithms and for different K on 3.6 GHz Xeon Gold 5122 processor, with 384 GB RAM, and Ubuntu 18 Linux OS. We recall that the time complexity of the greedy algorithm is proportional to NK, while the time complexity of the optimal algorithm is proportional to NK^2. The difference in processing speed between the two algorithms is therefore dependent on the size of K. The input value K is proportionally divided according to the number and sizes of separate maternal or paternal lines. Due to the rounding-off error the actual number of selected individuals K^* is usually smaller than K. This effect is more pronounced in pedigrees with large number of lines and a greater variation in lineage lengths, as is the case with real pedigrees. However, the processing is fast and a trial and error procedure should quickly achieve a desired K^*. Preprocessing time depends mainly on the pedigree size, and only slightly on the number of separate lineages.

The large Czech Holstein pedigree that was made available to us was anonymized, as a result we haven't been able to define the reference population according to the year of birth. The reference population was therefore set to be the entire pedigree. For the reasons of consistency, in our experiments we treated all of the test pedigrees equally.

The results presented in Table 1. show that the optimal method is significantly slower in some relevant cases. As the difference in the quality of output between two methods is negligible, we consider the greedy method as the method of choice.

To illustrate the small difference in the output quality between the optimal and greedy methods, in Table 2. we present scores (sums of distances between the selected individuals) for three representative cases. We see that the difference is very small, as in all of our experiments it was never larger than 0.01%.

Table 2. Quality scores for optimal and greedy methods, for three sample cases. The values in parenthesis are the differences from optimal results, given in percentages. Result for the method implemented in Magellan software is given only for the smallest sample, as that method is too slow for the larger pedigrees.

Method/Case	**Simulated, n = 1 M 50 Lineages, $K = 10^3$**	**Simulated, n = 20 M 500 Lineages, $K = 10^4$**	**Real $K = 3 \times 10^5$**
optimal	217,459	2,227,460	899,067
greedy	217,455	2,227,460	898,974
	$(-1.8 \times 10^{-3}\%)$	$(-8.5 \times 10^{-4}\%)$	$(-1.0 \times 10^{-2}\%)$
Magellan	217,448	N/A	N/A
	$(-5.1 \times 10^{-3}\%)$		

Incidentally, processing pedigrees with 20 million individuals in an acceptable time was not feasible with Magellan software. In the single experiment, we stopped the program when it did not conclude after 10 days of running. In another example, processing of a pedigree with one million individuals required over two days. Therefore, using Magellan software on large pedigrees is not realistic, and we have not included these results in Table 1. Furthermore, the heuristic method used in Magellan is different from the greedy method described in this paper and, besides being slower, it also produces a somewhat lower score. The example of this is presented in Table 2.

4. Conclusions

We have presented the first software that can perform the selection of candidates for sampling in real time out of medium-sized and large pedigrees, with regard to a single-gender lineage. The previous software for this purpose [4] could not process large pedigrees in realistic time. While very large pedigrees are rare, the main benefit of new algorithms is much faster processing of the medium-sized pedigrees. As an illustration, on pedigrees with sizes of approximately 10^5 individuals, the new algorithms are two orders of magnitude faster.

Based on the experimental results, we conclude that, out of two presented algorithms, the suboptimal algorithm is the best choice for practical usage. We expect this will benefit researchers dealing with the mitochondrial DNA and Y chromosome variability in a population.

Author Contributions: D.H.: Software, Validation, Writing—Review & Editing. I.K.: Software, Methodology, Formal analysis, Writing—Review & Editing. S.R.: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing—Original Draft, Writing—Review & Editing, Supervision, Project administration, Funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the Croatian Science Foundation grants IP-2018-01-7317 and IP-2018-01-8708, and European Regional Development Fund [KK.01.1.1.01.0009—DATACROSS].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We thank Ino Čurik and Luboš Vostrý for help with access to a large real pedigree and the useful comments.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Bogyo, T.P.; Porceddu, E.; Perrino, P. Analysis of sampling strategies for collecting genetic material1. *Econ. Bot.* **1980**, *34*, 160–174. [CrossRef]
2. Crossa, J.; Vencovsky, R. Basic Sampling Strategies: Theory and Practice. 2011. Available online: https://cropgenebank.sgrp.cgiar.org/images/file/procedures/collecting2011/Chapter5-2011.pdf (accessed on 26 January 2022).
3. Rosenberger, K.; Schumacher, E.; Brown, A.; Hoban, S. Proportional sampling strategy often captures more genetic diversity when population sizes vary. *Biol. Conserv.* **2021**, *261*, 109261. [CrossRef]
4. Ristov, S.; Brajkovic, V.; Cubric-Curik, V.; Michieli, I.; Curik, I. MaGelLAn 1.0: A software to facilitate quantitative and population genetic analysis of maternal inheritance by combination of molecular and pedigree information. *Genet. Sel. Evol.* **2016**, *48*, 65. [CrossRef] [PubMed]
5. Cubric-Curik, V.; Novosel, D.; Brajkovic, V.; Rota Stabelli, O.; Krebs, S.; Sölkner, J.; Šalamon, D.; Ristov, S.; Berger, B.; Trivizaki, S.; et al. Large-scale mitogenome sequencing reveals consecutive expansions of domestic taurine cattle and supports sporadic aurochs introgression. *Evol. Appl.* **2021**, 1–16. Available online: https://onlinelibrary.wiley.com/doi/pdf/10.1111/eva.13315 (accessed on 26 January 2022).
6. Nosková, A.; Přibyl, J.; Vostrý, L. Relationships between conformation traits and milk yield, lifetime production and number of lactations in Czech Holstein cows. In Proceedings of the Annual ICAR Conference 2019, Praha, Czech Republic, 21–25 June 2019.