Estimating population size via line graph reconstruction

Bjarni V. Halldórsson[1], Dima Blokh[2], and Roded Sharan[2]

(1) School of Science and Engineering, Reykjavík University, Menntavegur 1, 101 Reykjavik, Iceland.
Email: bjarnivh@ru.is
(2) Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.
Email: roded@post.tau.ac.il.

Abstract. We propose a novel graph theoretic method to estimate haplotype population size from genotype data. The method considers only the potential sharing of haplotypes between individuals and is based on transforming the graph of potential haplotype sharing into a line graph using a minimum number of edge and vertex deletions. We show that the problems are NP complete and provide exact integer programming solutions for them. We test our approach using extensive simulations of multiple population evolution and genotypes sampling scenarios. Our computational experiments show that when most of the sharings are true sharings the problem can be solved very fast and the estimated size is very close to the true size; when many of the potential sharings do not stem from true haplotype sharing, our method gives reasonable lower bounds on the underlying number of haplotypes. In comparison, a naive approach of phasing the input genotypes provides trivial upper bounds of twice the number of genotypes.

1 Introduction

A fundamental problem in population studies is the estimation of the size of the underlying haplotype pool. In these studies, such as genomewide association studies, a number of individuals are genotyped at some single nucleotide polymorphism (SNP) locations. Since we cannot observe the haplotypes of an individual, a common approach to the size estimation problem is to phase the genotype data, i.e., computationally predict the underlying haplotypes. However, haplotype phasing is a notoriously difficult problem [8] and can be optimally solved for small instances only [4].

Here we propose a novel approach that does not require the phasing of the haplotypes. It is based on starting from the easy-to-compute information on potential haplotype sharing between individuals. Specifically, we can test if two individuals have the potential to share a haplotype by checking if their genotypes are consistent with such sharing (i.e., whenever one of the genotypes is homozygous at a certain site, the other is homozygous to the same allele or heterozygous). This information can be encoded in a graph, known as the Clark consistency (CC) graph [7], where each individual (genotype) is represented by a node and an edge is added between two individuals if they can share a haplotype.

The line graph of a graph, is a new graph that has nodes for each edge in the root (original) graph and an edge between two nodes of the line graph if the corresponding edges are adjacent in the root graph. If data were perfect, i.e., the only observed potential sharings were true sharings, then the resulting CC graph would be a line graph whose root graph has haplotypes as nodes and edges connect pairs of haplotypes corresponding to the observed genotypes. The reconstruction of this root graph would then enable us to compute the haplotype population size. In practice, the graph is only close to being a line graph and contains "extra" edges

that do not represent true sharing. These extra edges are due to the fact that the genotypes are consistent with each other while there is no common shared haplotypes. For reasonable population structure (as reflected in the simulations presented below), we expect such sharing of genotypes to be rare. Thus, our goal is to find a smallest number of edge deletions that will make the CC graph a line graph. Once we arrive at a line graph we can reconstruct its root graph, thereby getting an estimate of the number of underlying haplotypes, as well as predictions of which individuals share a haplotype (those pairs that remain connected by an edge). We also consider a variant of the line graph deletion problem where node deletions are allowed; those correspond to cases where there are significant genotyping errors or missing data in one of the individuals or genotype pools.

CC graphs may also be determined from data generated using other technologies or experimental protocols, including pooled sequencing [12]. In pooled sequencing, pools of DNA are constructed, each containining the DNA of multiple individuals with each individual typically belonging to multiple pools. Each pool is then genotyped and a conflated genotype for all the individuals in the pool is constructed. To create a CC graph we add an edge between two individuals if they can share a haplotype, that is, if for every pair of pools containing these individuals, one in each pool, the two pools are consistent with sharing a haplotype.

We study the complexity of the resulting line graph modification problems. We observe that both problems (edge- and vertex-deletion) are NP-complete and provide polynomial integer programming formulations to solve them. We test our approach using extensive simulations of multiple population evolution and genotypes sampling scenarios. Our computational experiments show that when most of the sharings are true sharings the problem can be solved very fast and the estimated size is very close to the true size. In cases where many of the potential sharings do not stem from true haplotype sharing, due to spurious edge or node insertions, our method gives reasonable lower bounds on the underlying number of haplotypes. In comparison, a naive approach of phasing the input genotypes provides trivial upper bounds of twice the number of genotypes.

2 Preliminaries

Let $G = (V, E)$ be a graph with a set V of n vertices and a set E of m edges. The line graph of G, denoted by $L(G)$, is constructed by having a node in $L(G)$ for each edge in G and an edge between two nodes of $L(G)$ if the corresponding edges are adjacent in G.

2.1 Line graph and CC graphs

If $L(G)$ is the line graph of G, we refer to G as the *root graph* of $L(G)$. Line graphs have been studied by a number of authors. Whitney [16] showed that, apart from a single exception, the root graph of a line graph is unique. Lehot [10] and Roussopoulos [13] have independently given linear-time algorithms for detecting whether a graph is a line graph and outputting its root graph. Van Rooij and Wilf [15] gave a characteriziation of line graphs in terms of nine forbidden subgraphs. The characterization can be stated as follows; A triangle in a graph is

even if every other node is adjacent to 0 or 2 nodes in the triangle; it is *odd* otherwise. A graph is a line graph iff it contains no two odd triangles that share an edge is claw-free (a claw is a node connected to three nodes not connected to each other). This characterization can also be stated as a list of nine distinct subgraphs that are forbidden from the line graph. Each one of the nine different subgraphs has at most six nodes and ten edges.

A key component of our approach is the graph of potential sharing of haplotypes between individuals. This graph, called the Clark consistency (CC) graph, was first suggested in the context of a method for haplotype phasing by Andrew Clark [5]. Given the genotypes of a set of individuals, the CC graph has one node for each individual and an edge between two individuals if their genotypes are consistent with sharing a haplotype, i.e., if for every site where one of the individuals is homozygous, the other individual is either homozygous for the same allele or heterozygous. As the haplotypes of individuals that are homozygous for the whole region being considered are easily determined, we assume that every given individual has two different haplotypes (i.e., it is heterozygous for at least one of the genotyped markers) and no two individuals have the same pair of haplotypes.

As we show below, CC graphs and line graphs are closely related. Let H be the CC graph of a given set of genotypes. We say that H is *allelable* if there is an assignment of pairs of colors to its nodes so that every two adjacent nodes share exactly one color and every two non-adjacent nodes have distinct color sets. Under this assignment every color represents a haplotype or an allele in the population. The number of colors is the estimated number of genotypes. The following observation stands at the basis of our computational approach.

Lemma 1. *H is allelable iff it is a line graph.*

Proof. If H is allelable then construct a graph G in which every allele of H represents a node and edges connect alleles that are paired together in the nodes of H. It is easy to see that H is the line graph of G.

Conversely, if $H = L(G)$ is a line graph then one can assign distinct labels to the nodes of the corresponding root graph G and use these labels to label the edges of G and, hence, the nodes of H. This assignment clearly shows that H is allelable.

□

Lemma 1 implies that a set of haplotypes providing a valid phasing of genotypes will always form a line graph. The converse, however, need not be true as the SNP data of the original genotypes are not encoded in our formulation.

2.2 Problem definition and its complexity

As the input data may have more edges than the underlying line graph, we do not expect real data to yield line graphs. Rather it is desirable to transform a given CC graph into a line graph using a minimum number of node and edge deletions. In the following we formulate three versions of the problem.

Problem 1. Given a graph $G = (V, E)$, find a line graph $G' = (V, E')$, such that $E' \subseteq E$ and $\mid E - E' \mid$ is minimized.

Problem 2. Given a graph $G = (V, E)$, find a line graph $G' = (V', E)$, such that $V' \subseteq V$ and $\mid V - V' \mid$ is minimized.

We also consider a combined problem where both nodes and edges may be removed from a graph:

Problem 3. Given a graph $G = (V, E)$ and a constant α, find a line graph $G' = (V', E')$, such that $E' \subseteq E$, $V' \subseteq V$ and $\mid E - E' \mid + \alpha \mid V - V' \mid$ is minimized.

Yannakakis [17] has shown that the problems of deleting a minimum number of edges or nodes of a graph in order to make it into a line graph are both NP-hard.

As the problems can be formulated as a problem of avoiding particular subgraphs which have at most 6 nodes and 10 edges, by [2] both problems are fixed parameter tractable and can be solved in time $\max\{m, n\}O(10^i)$ and $\max\{m, n\}O(6^j)$, respectively, where i (resp., j) is the minimum number of edges (resp., vertices) that need to be deleted. Using the techniques of Niedermeier and Rossmanith [11], the node deletion variant can be solved in $\max\{m, n\}O(5.19^j)$ time.

Problem 2 can be formulated as a hitting set problem, where all the sets that need to be hit are of size $4, 5$ or 6. Trevisan [14] showed that the hitting set problem, when the size of the input sets is bounded by $B \geq 2$ ($B = 6$ in our case), is hard to approximate to within $B^{-\frac{1}{19}}$. The fact that all sets are of size at most six also implies a 6-approximation algorithm for the line graph node deletion problem [6].

3 Integer programming formulation

Here we provide integer programming formulations for Problems 1, 2 and 3. The formulations rely on the characterization given in Lemma 1. We let $[i, j]$ represent $\{i, i + 1, ..., j\}$.

3.1 Edge deletions

Let $G = (V, E)$ be the input CC graph. Our basic program has the following sets of binary variables: (i) A variable d_e for every edge $e \in E$, where $d_e = 1$ iff e is deleted. (ii) A variable $x_{v,j}$ for every node $v \in V$ and every possible allele $j \in [1, 2n]$, where $x_{v,j} = 1$ iff individual v has allele j. (iii) A variable $s_{e,j}$ for every edge $e \in E$ and every possible allele $j \in [1, 2n]$, where $s_{e,j} = 1$ iff the two endpoints of e share allele j. It is formulated as follows:

$$
\begin{array}{lll}
\min & \sum_e d_e & \\
s.t. & \sum_{j=1}^{2n} x_{v,j} = 2 & \forall v \in V \\
& x_{v,j} + x_{u,j} \leq 1 & \forall (u, v) \notin E, j \in [1, 2n] \\
& \sum_{j=1}^{2n} s_{e,j} = 1 - d_e & \forall e \in E \\
& x_{u,j} + x_{v,j} - 1 \leq s_{e,j} & \forall e = (v, u) \in E, j \in [1, 2n] \\
& s_{e,j} \leq x_{u,j} & \forall j \in [1, 2n], u \in e \\
& d_e, x_{v,j}, s_{e,j} \in \{0, 1\} & \forall e \in E, v \in V, j \in [1, 2n]
\end{array}
$$

In this formulation, the first constraint ensures that each node is assigned two distinct colors; the next two constraints ensure that non-adjacent nodes (before or after edge deletion) do not share a color; and fourth and fifth constraints ensure that the edge sharing variables are consistent with the node coloring variables. A potential problem with the formulation, however, is that different permutations of the colors yield equivalent solutions. To tackle this problem, we use the symmetry breaking techniques of [3]. Specifically, we start by ordering the nodes from 1 through n. Each node "owns" two colors: node i owns colors $(2i-1)$ and $2i$. The fact that $x_{v,j} = 1$, where $j = 2v-1$ or $j = 2v$ means that v is the first node to use color j. v is required to use color $2v-1$ before it uses color $2v$.

Overall, node v has access to colors $1, 2, \ldots, 2v$ and can use some of the previously used colors or be the first ordered node to use new colors. These restrictions can be represented by the following additional constraints:

$$\begin{aligned} x_{v,2v} &\leq x_{v,2v-1} \quad \forall v \in [1,n] \\ x_{j,2i-1} &\leq x_{i,2i-1} \; \forall j \in [i+1,n] \\ x_{j,2i} &\leq x_{i,2i} \; \forall j \in [i+1,n] \end{aligned}$$

We observe that a number of the variables in the above formulation can be automatically set to 0 and removed from the formulation. In particular, if $(i,j) \notin E, i < j$ then $x_{j,2i} = x_{j,2i-1} = 0$, as j and i cannot share a color when they do not share an edge. If $e = (u,v)$, we also note that $s_{e,j} = 0$ unless both u, v can be colored with color j, that is if $(v, \left\lfloor \frac{j+1}{2} \right\rfloor), (u, \left\lfloor \frac{j+1}{2} \right\rfloor) \in E$.

3.2 Node deletions

If a graph does not contain one of the forbidden substructures then the graph is alleleable. Otherwise, a vertex has to be removed from each of the forbidden subgraphs. Our algorithm relies on listing all occurrences of the forbidden substructures and then solving the hitting set problem on the set of forbidden substructures.

We observe that the deletion of a node will not create a new forbidden substructure. The node deletion problem can thus be formulated as an integer program by listing all forbidden substructures and removing one node from each one of them. We let t_n be a binary variable representing whether node i is removed and denote by $\mathcal{F}$ the set of all forbidden induced subgraphs of a line graph. Then the program is formulated as follows:

$$\begin{aligned} \min \quad & \textstyle\sum_n t_n \\ & \textstyle\sum_{n \in F} t_n \geq 1 \; \forall F \in \mathcal{F} \\ & t_n \in \{0,1\} \;\; \forall n \in V \end{aligned}$$

3.3 Edge and node deletions

In real data, both genotype errors and spurious sharing relations may happen at the same time, leading to a combined node- and edge-deletion problem. We define d,x, s and t as before. We define the variable r_e as a boolean variable representing that an edge has been removed, which occurs when either one of its adjacent nodes or the edge itself is deleted. We let α be the relative cost of node deletion with respect to edge deletion. The combined program is as follows:

$$
\begin{array}{lcl}
\min & \sum_e d_e + \alpha \sum_n t_n & \\
s.t. & \sum_{j=1}^{2n} x_{v,j} = 2 - 2t_v & \forall v \in V \\
& x_{v,j} + x_{u,j} \leq 1 & \forall (u,v) \notin E, j \in [1,2n] \\
& \sum_{j=1}^{2n} s_{e,j} = 1 - r_e & \forall e \in E \\
& x_{u,j} + x_{v,j} - 1 \leq s_{e,j} & \forall e = (v,u) \in E, j \in [1,2n] \\
& s_{e,j} \leq x_{u,j} & \forall j \in [1,2n], u \in e \\
& r_e \leq t_v + t_u + d_e & \forall e = (u,v) \in E \\
& d_e \leq r_e & \forall e \in E \\
& t_u \leq r_e & \forall u \in e, e \in E \\
& d_e, t_v, x_{v,j}, s_{e,j}, r_e \in \{0,1\} & \forall e \in E, v \in V, j \in [1,2n]
\end{array}
$$

We further augment this integer program with the symmetry breaking techniques described in Section 3.1.

4 Experimental results

We comprehensively test our algorithm for deleting edges and vertices under two simulated scenarios, corresponding to a bottleneck population isolate and a population that has continuously undergone recombination and mutation. For the population isolate we show that we can almost fully recover the haplotype structure from the sharing information alone. For the population that has undergone continuous recombination and mutation we get a bound on the number of haplotypes that is close to the true number of haplotypes in the population. Our experiments further reveal that the occurrence of genotypes showing a large degree of sharing does not materially affect our ability to estimate the number of haplotypes in the remaining population. The computational experiments were done using CPLEX 12, making use of a single CPU processor core with 4GB of memory.

4.1 Bottleneck population

Haplotypes that are shared across a long region are with high probability identical by descent, i.e. they are the result of the genetic material of a single forefather being passed to a

number of his descendents. Some of the haplotypes, however, will be identical by state only, i.e., the haplotypes are the same but cannot be traced to a single common forefather. The probability of identical by state sharing decreases rapidly with the length of the haplotype being considered. The probability of identity by state sharing depends on the size of the ancestral population. We simulate graphs that might occur from the detection of identity by state sharing.

We use Hudson's ms simulator [9] to simulate realistic genotype populations. We assume that there is an initial small bottleneck population that then rapidly expanded. We simulate genotypes for the initial population and then generate the current population as a random sample with replacement from this initial population. The parameters for the simulation are chosen such as to sample approximately a 5 cM locus, or 3027 SNPs, a size for which it is reasonable to expect that no recombinations would take place in the region between two individuals being studied. We consider two sets of experiments for this scenario. In the first experiment we vary the size of the population but keep the number of times that each haplotype is sampled on expectation fixed. In the second experiment we vary the number of times that each haplotype is sampled on expectation and keep the size of the population fixed.

Genotypes sampled	# edges	# edges removed	Estimated # haplotypes	True # haplotypes	Compute time (s)
25	21	0	33	33	0
50	46	1	64	63	0.04
75	74	0	95	95	0.01
100	105	0	125	126	0.02
125	129	2	154	153	0.05
250	294	1	300	300	0.07
500	634	3	589	594	0.29

Table 1. Performance evaluation. Genotypes are generated by sampling with replacement two haplotypes at a time from a population of size twice the number of genotypes. The size of this population is varied. The estimated and true numbers of haplotypes both refer to the set of haplotypes underlying the sampled genotypes.

In Table 1 we fix the number of times that a haplotype is sampled on average as 1 and we vary the size of the population, which we present as the number of haplotypes in the ancestral population. It can be seen that the number of edges grows roughly linearly with the size of the population being sample. Our simulations show that the number of estimated haplotypes is in close agreement with the true number of haplotypes in the sample. As some of the haplotypes are sampled multiple times the number of haplotypes in the sample is typically smaller than the actual population size. Notably, all instances are solved very fast (less than a second).

In Table 2 we look at an initial bottleneck population of 100 haplotypes while varying the number of genotypes sampled from this population from 25 to 125. This implies that each haplotype in the ancestral population is sampled between 0.5 and 2.5 times (on average). As the sample size increases, our estimate of the number of haplotypes tightly follows the true number of haplotypes sampled from the population, with both approaching the actual

# Genotypes	Coverage	# edges	# edges removed	Estimated # haplotypes	True # haplotypes	Compute time (s)
25	0.5	17	0	37	37	0.00
50	1	46	1	64	63	0.04
75	1.5	98	0	82	82	0.04
100	2	186	0	87	88	0.02
125	2.5	301	0	92	92	0.09

Table 2. Performance evaluation with respect to a sample drawn from an initial population of 100 haplotypes. The number of individuals drawn from the sample is varied.

population size of 100. We observe that the number of edges in the graph grows roughly as the square of the sample size. The CPU time used for all these instances is minimal.

4.2 Recombinant population

The second scenario that we simulate is a population that is undergoing mutation and recombination. We fix mutation rate at 10^{-9} per generation and recombination rate between two adjacent basepairs at 10^{-8} per generation. In all of our experiments we simulate a 1MB region, varying the number of individuals in the ancestral population (N_0) and the number of individuals sampled in the current population.

N_0	# SNPs	# genotypes	# edges	# edges removed	Estimated # haplotypes	True # haplotypes	Compute time (s)
500	116	100	484	245	83	144	1216
1000	231	100	235	105	114	172	120
2500	584	100	23	0	178	189	0.0
5000	1215	100	15	0	185	195	0.0
10000	3027	500	135	1	880	894	5

Table 3. Performance evaluation for a recombinant population while varying population size.

First, we vary the size of the initial population, while leaving the number of individuals considered mostly constant (see Table 3). We observe that when the size of the initial population grows the probability that two individuals can share a haplotype decreases rapidly. Many of the edges observed in these graphs are due to sharing between genotypes that are not due to the sharing of haplotypes. The graphs being considered are therefore far from being line graphs and many edges need to be removed in order to make them into ones. Nevertheless, we are able to give estimates of the number of haplotypes in a population even in this setting.

The quality of our estimate is, not surprisingly, dependent on the number of edges that need to be removed to create a line graph. Our estimated number of haplotypes is consistently smaller than the true number of haplotypes, but the number of estimated haplotypes is never below 57% of the true number of haplotypes. Similar conclusions can be drawn from the second set of experiments in which we fix the size of the initial population and vary the size of the sample that we draw from that population (see Table 4).

N_0	# SNPs	# genotypes	# edges	# edges removed	Estimated # haplotypes	True # haplotypes	Compute time (s)
1000	193	50	38	5	73	87	5
1000	231	100	235	105	114	172	120
1000	289	200	738	374	170	287	300

Table 4. Performance evaluation on a recombinant population while varying sample size.

4.3 Combined node and edge deletions

In our final set of experiments we simulate a scenario in which there are genotypes showing a high degree of excess sharing of haplotypes. To this end, we focus on the data set of 50 genotypes presented in Table 4, and designate increasing subsets of the genotypes as being partially observed, i.e., only a subset of their markers is observed supposedly due to failure or otherwise missing data.

# Partial genotypes	# Extra edges	α	# genotypes removed	# partial genotypes removed	# edges removed	Estimated # haplotypes	True # haplotypes
5	40	1.5	5	4	1	68	78
5	40	4	2	2	8	68	78
10	70	1.5	10	9	1	63	71
10	70	4	4	3	18	63	71
15	102	1.5	13	9	0	55	62
15	102	4	7	6	14	59	62

Table 5. Performance evaluation in a population where some of the genotypes are only partially observed.

In more detail, in Table 5 we modify a graph consisting of 50 genotypes, consisting of 38 edges where 5 edges have to be removed to transform the graph into a line graph. We let $5, 10$ or 15 of the genotypes be partially observed, i.e., genotyped at only at 30% of their markers. We show the number of edges added to the graph due to these partial observations. Next, we apply our combined node and edge deletion algorithm and use two settings for the alpha parameter controlling the relative penalty of node vs. edge deletion. The first value that we use, $\alpha = 1.5$, prefers a node deletion whenever more than one of the node's adjacent edges has to be deleted. The second value of 4 allows up to 4 edge deletions before the node deletion is preferred. In each setting we provide the the number of edges and genotypes removed and the number of removed genotypes that were partially observed. We evaluate the size estimate given by the algorithm against the true number of haplotypes in the data set without counting the haplotypes in the partially observed genotypes.

We observe that the occurrence of the partial genotypes does not appear to materially affect our estimate of the number of haplotypes in the remaining population. We also observe that the largest number of nodes removed in each experiment are from the subset corresponding to the partially observed genotypes. When $\alpha = 1.5$, $60 - 90\%$ of the partially observed genotypes are removed and less then 12% of the other genotypes. When $\alpha = 4$, $30 - 40\%$ of the partial genotypes are removed and less then 3% of the other genotypes. All computations presented in Table 5 finished in under one second.

4.4 Comparison to a phasing-based estimation

Apart from the preprocessing, the complexity of our method does not depend on the number of genotyped markers. In contrast, many haplotype phasing methods are not able to handle the large number of markers dealt with in our approach [4]. We thus opted to compare our method to phasing-based estimates derived from the application of the BEAGLE phaser [1] to our data. Surprisingly, in all the simulated settings, the number of haplotypes estimated by BEAGLE was twice the number of genotypes, which is a trivial upper bound for the number of haplotypes in a population.

5 Conclusions

We show that the problem of assigning alleles to individuals when only information about the sharing of alleles between individuals is known is equivalent to the problem of determining whether a graph is a line graph. When sharing information is not perfect we give polynomial size integer programming algorithms for determining allele sharing.

Researchers are frequently interested in knowing not only the number of haplotypes for a particular instance but also the haplotypes themselves. While the method presented here does not address the problem of determining haplotypes, we hope it may be used in conjunction with or as a part of such a method. In particular, one could modify our approach to remove edges in the graph such that the resulting graph is a line graph with a minimum number of haplotypes. The line graph minimizing the number of haplotypes will always provide a lower bound on the true number of haplotypes in the sample; thus, the approach presented here could provide the basis for a constraint generation technique for minimizing the number of haplotypes.

The model considered here does not deal with noise due to genotype miscalls. Genotyping errors may lead to haplotypes not sharing an edge when they in fact should. There are two ways in which this problem can be handled by our framework: Edges can be added when the genotypes are compatible in all but a fixed number of error locations, or we may allow for edge insertions, possibly with weights related to the number of errors that will need to be corrected for such edges to occur. Our integer programming formulation can be easily extended to allow for such edge insertions, solving the more general problem of editing a graph (by edge insertions and deletions) to become a line graph.

Acknowledgments

RS was supported by a research grant from the Israel Science Foundation (grant no. 241/11).

References

1. B. L. Browning and S. R. Browning. A unified approach to genotype imputation and haplotype-phase inference for large data sets of trios and unrelated individuals. *American Journal of Human Genetics*, 84(2):210–223, 2009.
2. L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58:171–6, 1996.

3. M. Campelo, V. Campos, and R. Correa. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156(7):1097–1111, 2008.
4. D. Catanzaro, A. Godi, and M. Labbé. A class representative model for pure parsimony haplotyping. *INFORMS Journal of Computing*, 22(2):195–209, 2009.
5. A. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution*, 7:111–122, 1990.
6. S. Even and R. Bar-Yehuda. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.
7. B. V. Halldórsson, D. Aguiar, R. Tarpine, and S. Istrail. The Clark Phaseable Sample Size Problem: Long-Range Phasing and Loss of Heterozygosity in GWAS. *Journal of Computational Biology*, 18(3):323–333, Mar. 2011.
8. B. V. Halldórsson, V. Bafna, N. Edwards, R. Lippert, S. Yooseph, and S. Istrail. A survey of computational methods for determining haplotypes. In *Computational Methods for SNPs and Haplotype Inference (LNCS 2983)*, pages 26–47. Springer, 2004.
9. R. R. Hudson. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002.
10. P. G. H. Lehot. An optimal algorithm to detect a line graph and output its root graph. *J. ACM*, 21:569–575, October 1974.
11. R. Niedermeier and P. Rossmanith. An efficient fixed-parameter algorithm for 3-hitting set. *Journal of Discrete Algorithms*, 1(1):89–102, 2003.
12. S. Prabhu and I. Pe'er. Overlapping pools for high-throughput targeted resequencing. *Genome Research*, 19:1254–61, 2009.
13. N. Roussopoulos. A max(m,n) algorithm for determining the graph H from its line graph G. *Information Processing Letters*, 2:108–112, 1974.
14. L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 453–461. ACM, 2001.
15. A. Van Rooij and H. Wilf. The interchange graphs of a finite graph. *Acta Math. Acad. Sci. Hungar.*, 16:263–269, 1965.
16. H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54:150–162, 1932.
17. M. Yannakakis. Node-and edge-deletion NP-complete problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, STOC '78, pages 253–264, New York, NY, USA, 1978. ACM.