BIOINFORMATICS ORIGINAL PAPER Vol. 21 no. 10 2005, pages 2463–2468
doi:10.1093/bioinformatics/bti373

Genetics and population analysis

Partition-distance via the assignment problem

Dmitry A. Konovalov*, Bruce Litow and Nigel Bajema

School of Information Technology, James Cook University, Townsville, QLD 4811, Australia

Received on November 30, 2004; revised on February 15, 2005; accepted on March 2, 2005
Advance Access publication March 3, 2005

ABSTRACT

Motivation: Accuracy testing of various pedigree reconstruction methods requires an efficient algorithm for the calculation of distance between a known partition and its reconstruction. The currently used algorithm of Almudevar and Field takes a prohibitively long time for certain partitions and population sizes.

Results: We present an algorithm that very efficiently reduces the partition-distance calculation to the classic assignment problem of weighted bipartite graphs that has known polynomial-time solutions. The performance of the algorithm is tested against the Almudevar and Field partition-distance algorithm to verify the significant improvement in speed.

Availability: Computer code written in java is available upon request from the first author.

Contact: dmitry.konovalov@jcu.edu.au

INTRODUCTION

Luikart and England (1999) highlighted how the availability of suitable genetic markers, in particular, microsatellite markers created a need for accurate pedigree analysis methods in order to analyze the flood of empirical studies. Jones and Arden (2003) reviewed a number of such methods which are implemented as computer software and readily available. Since the publication of the review of Jones and Arden (2003), another method named the Descending Ratio and corresponding computer program KinGroup have become available based on the overall likelihood (Konovalov *et al*., 2004). However, from a practitioner's point of view, any new and especially statistical method should provide the accuracy estimation of the partitioning results. Butler *et al*. (2004) used a number of accuracy related parameters, with one of the most important being the distance between two partitions as defined by Almudevar and Field (1999)—the minimum number of individuals that need to be removed from each partition in order to leave the remaining partitions equal. This distance is equivalent to the number of individuals who have to be moved to a different cluster in order to reproduce the given partition. An elegant and simple-to-implement algorithm to calculate the partition-distance was reported by Almudevar and Field (1999) (hereafter the AF algorithm) and is still in use, e.g. by Butler *et al*. (2004).

Butler *et al*. (2004) used the AF algorithm in their accuracy study of four full sibship reconstruction algorithms. However, they reported only a relatively small number (six) of trials for each choice of the family structure, allelic frequencies, number of loci and alleles. In contrast, Beyer and May (2003) ran 100 trials for each family structure to estimate the accuracy of their graph-theoretical approach to the partition of full-sib families. Yet, for example, the users of the popular Kinship software program by Goodnight and Queller (1999) are accustomed to an even higher number of simulations. Typically, thousands of simulations would be expected to be computable within acceptable time on consumer grade computers. Such simulations can not be generically pre-computed as each individual case has unique family and genetics structures and there is a dramatic difference in the reconstruction accuracy based on different allelic frequencies and family size distributions as demonstrated by Beyer and May (2003). These simulations must be calculated 'on-the-fly' to assign a specific confidence level to the result.

Butler *et al*. (2004) suggested and applied a comprehensive set of tests that could be used as a benchmark for a new method such as the Descending Ratio by Konovalov *et al*. (2004). Our attempts to compare against the accuracy results of the four methods led us to the realization that the AF algorithm is prohibitively slow in some cases, e.g. calculating the partition-distances when 50 and 200 unrelated individuals were partitioned using a low number of loci. Jones and Arden (2003) highlighted the importance of the pedigree reconstruction methods (parentage assignment included) being readily available to practitioners. Hence even though the AF algorithm could be used in one-off academic simulations, it could be unacceptably slow to be included in software for general use such as the KinGroup program by Konovalov *et al*. (2004).

This paper focuses on the distance calculation arising in pedigree reconstruction methods. We derived an algorithm that is comparable to the AF algorithm in being easy to implement but having the significant advantage of being much faster with a known polynomial-time complexity measure. We achieved that by reducing the partition-distance problem to the classic assignment problem [see for example Papadimitriou and Steiglitz (1998)]. Our reduction is similar to the reduction given in Gusfield (2002), where it was first shown that the partition-distance can be computed by a reduction to the assignment problem, with the same polynomial time bound as in this paper.

ALGORITHM

Following the notation and terminology of Gusfield (2002) we define a population of individuals as a set of p-elements $P = \{1, \ldots, p\}$, where p is the population size. A given population can be partitioned into clusters of individuals based on a required characteristic, e.g. kin groups in Konovalov *et al*. (2004). Each cluster is a non-empty subset of P or an empty set if required for convenience of derivation or calculation. A partition of the given population is a set of mutually exclusive clusters whose union is the population. Such a definition

*To whom correspondence should be addressed.

of a partition allows handling a number of reconstruction problems, including the full sibship assignment. Two possible partitions A and B (with corresponding partition sizes r and s) are denoted as

$$A(r) = \{a_1, \ldots, a_r\}, \quad B(s) = \{b_1, \ldots, b_s\}. \tag{1}$$

Without losing generality, the smaller partition is labeled as A, hence $r \leq s \leq p$. Given two partitions A and B, Almudevar and Field (1999) defined the partition-distance $D(A, B)$ as the minimum number of elements that need to be removed from both partitions to make them identical.

We start the reduction by representing any partition $A(r)$ of the set $P = \{1, \ldots, p\}$ as a collection of length p binary strings or bit sets such that

$$\bigcup_{i=1}^{r} a_i = I, \tag{2}$$

where I is the all-ones string of length p. We use the same notation a_i for both a set and a binary string. Following Almudevar and Field (1999), the partition-distance problem $D(A, B)$ can then be recast in terms of the strings as follows. Find a string C such that $|C|$ is least and

$$A \cap \overline{C} = B \cap \overline{C}, \tag{3}$$

where $\overline{C}$ is the complementary string to C, and $|C|$ is the cardinality of C corresponding to the number of 1 bits in the string. After finding the minimum cardinality string C, the partition-distance becomes

$$D(A, B) = \min |C|. \tag{4}$$

We proceed by looking for an expression for C in such a form that $|C|$ could be directly calculated from the A and B bit sets. Using the fact that any two strings a and b are equal iff (if and only if) $a \oplus b = 0$, where $\oplus$ denotes the XOR symmetric difference operator, and expanding A via

$$A \cap \overline{C} = \bigcup_i a_i \cap \overline{C}, \tag{5}$$

Equation (3) becomes

$$\bigcup_{ij} x_{ij}(a_i \oplus b_j) \cap \overline{C} = 0, \tag{6}$$

and x_{ij} is a permutation matrix allowing for arbitrary combination of clusters from partitions A and B such that each cluster is used only once. The union in Equation (6) is done over the maximum partition sizes of A and B by creating empty partitions if required. Equation (6) in fact defines C since $Z \cap \overline{C}$ iff $Z \subseteq C$. Thus we obtain an explicit expression for C,

$$C = \bigcup_{ij} x_{ij}(a_i \oplus b_j). \tag{7}$$

However Equation (7) is not directly suitable for cardinality calculation as each of its terms may be overlapping thus giving only an upper bound to $|C|$. In order to create a sum of non-overlapping terms, we use the partition orthogonally property from Equation (2),

$$C = \bigcup_i a_i \cap \bigcup_{i'j} x_{i'j}(a_{i'} \oplus b_j). \tag{8}$$

Using the clusters' orthogonality

$$a_i \cap a_{i'} = a_i \delta_{ii'}, \tag{9}$$

$$a_i \cap \bar{a}_{i'} = a_i(1 - \delta_{ii'}) \tag{10}$$

and the XOR property

$$a \oplus b = (a \cap \overline{b}) \cup (\overline{a} \cap b), \tag{11}$$

Equation (8) becomes

$$C = \left(\bigcup_{ij} x_{ij}(a_i \cdot \overline{b}_j) \right) \cup \left(\bigcup_{ij} a_i \bigcup_{i' \neq i} x_{i'j} b_j \right). \tag{12}$$

Due to the clusters' orthogonality property, the union of all but one cluster equals the complement of the excluded cluster,

$$\bigcup_{i' \neq i} x_{i'j} b_j = x_{ij} \overline{b}_j. \tag{13}$$

Equation (12) is then reduced to

$$C = \bigcup_{ij} x_{ij}(a_i \cap \overline{b}_j). \tag{14}$$

By the derivation, all terms in Equation (14) are now non-overlapping; hence the cardinality can be directly calculated via

$$|C| = \sum_{ij} x_{ij} |a_i \cap \overline{b}_j|. \tag{15}$$

Thus the partition-distance is reduced to

$$D(A, B) = \min \sum_{ij} x_{ij} |a_i \cap \overline{b}_j|, \tag{16}$$

where x_{ij} is a set of assignment-constraining variables such that only one element in each row or column has the value of 1 with the rest being zeros. Equation (16) is an instance of the standard weighted assignment problem as defined, for example, by Papadimitriou and Steiglitz (1998), where $|a_i \cap \overline{b}_j|$ is the cost matrix. The reduction of Gusfield (2002) in terms of this paper becomes

$$D(A, B) = p - \max \sum_{ij} x_{ij} |a_i \cap b_j|. \tag{17}$$

We implemented both Equations (16) and (17) formulations and verified that they produced identical distances in all cases. The independently derived formulations achieve methodologically the same result, verifying the existence of the assignment problem reduction. In hindsight, it could be shown that Equation (16) could have been derived from Equation (17) directly.

Table 1. Sample workings of the algorithm[a]

$P = \{abcdef\}$ = '111111'	$b_1 = \{abf\}$ = '110001', $\bar{b}_1$ = '001110'	$b_2 = \{de\}$ = '000110', $\bar{b}_2$ = '111001'	$b_3 = \{c\}$ = '001000', $\bar{b}_3$ = '110111'
$a_1 = \{abdf\}$ = '110101'	$\lvert 000100 \rvert = \mathbf{1}$	$\lvert 110001 \rvert = 3$	$\lvert 110101 \rvert = 4$
$a_2 = \{ce\}$ = '001010'	$\lvert 001010 \rvert = 2$	$\lvert 001000 \rvert = \mathbf{1}$	$\lvert 000010 \rvert = 1$
$a_3 = \{\}$ = '000000'	$\lvert 000000 \rvert = 0$	$\lvert 000000 \rvert = 0$	$\lvert 000000 \rvert = \mathbf{0}$

[a] A cost matrix from Equation (16) with the sample workings of the algorithm when applied to the example from Gusfield (2002). In this example the partition-distance is 2 (a possible assignment selection is highlighted in bold).

Sample application of the algorithm

A sample working is shown in Table 1 for the example from Gusfield (2002) to illustrate the algorithm and its notations. Clusters of individuals from the population P with $p = 6$ individuals a, b, c, d, e and f are represented by the set notations as well as bit strings. Each individual is assigned a position in the bit string which is set to 1 if that individual is present in the represented cluster. The cost matrix is calculated using Equation (16) by multiplying each a-row and b-column, with the intermediate bit strings shown in Table 1 before the cardinality operation is applied. The resulting minimum assignment problem could be solved manually in this simple example. The solution is the minimum sum of all individual terms where each row is paired with only one column.

Implementation

The minimum weighted assignment problem has a number of known polynomial-time solutions. We chose the method implemented in LEDA by Mehlhorn and Näher (1999). We have converted LEDA C++ template code to java besides using it directly (via the JNI-Java Native Interface) to run all performance testing. Once the cost matrix from Equation (16) is calculated, the resulting assignment problem is solved using LEDA (both native C++ code and our java conversion). We have also implemented the AF algorithm in java for comparison. We tested the following three methods. The first is this paper's algorithm backed by LEDA native C++ implementation, the second replaces C++ with our java implementation and the third being the slightly improved (see Appendix) AF algorithm (in java). All three methods attain identical distances in all cases, cross-validating their implementations. We chose to implement and test the discussed algorithms in java due to the continuing work within the java-based KinGroup project by Konovalov *et al*. (2004).

RESULTS

Given two partitions A and B, the effective partition sizes are defined As r' and s' [note the corresponding r and s in Equation (1)] after all common clusters are removed. The maximum effective partition size is defined as $n = \max(r', s')$. It follows from Equation (16) that the assignment problem-based algorithm's running time is solely a function of n, not the population size p. In some cases a partition may have a known number of individuals per cluster. Such cases will be denoted as

$$A(r, m), \tag{18}$$

where r is the partition size or the number of clusters and m is number of elements per cluster.

Simulation test R1

The first simulation test is arranged to emulate possible outcomes of accuracy testing in Butler *et al*. (2004) and Konovalov *et al*. (2004) for unrelated individuals with insufficient genetic information (small number of loci and/or alleles). Partition A was created to be $A(r, 1) = \{a_1, \ldots, a_r\}$ such that each cluster a_i contained only one element giving $p = r$. Partition B was randomly created using the following algorithm:

(1) Starting from all p-individuals being unassigned, randomly select an individual from the pool of unassigned individuals.

(2) Randomly select either an existing or a new cluster. Assign the individual to the cluster. For example, given three existing clusters, there will be a probability of 0.25 for the individual been assigned to one of the existing clusters or to a newly created cluster.

(3) Continue steps (1) and (2) until all individuals are assigned.

The structure of partition B was chosen for computational convenience to exaggerate the misclassification effect in order to obtain a wide spread of n values faster. The biologically realistic results of the full sibship reconstruction achieve as high partition-distance values as 30; for example when Butler *et al*. (2004) reconstructed 50 unrelated individuals described by four loci with four equifrequent alleles in each locus. In the context of this paper, the 50 unrelated individuals is an instance of the $A(50, 1)$ partition while the partition-distance of 30 is equivalent to the effective partition size of $n = 30$ [such direct equivalence is true for the $A(r, 1)$ structures]. The distance of 30 takes approximately 20 times longer to compute via the AF algorithm than this paper's algorithm (Fig. 1).

Figure 1 compares this paper's algorithm backed by java and C++ versions of the LEDA's assignment method mainly to illustrate that even though the java version is slower it is comparable in its behavior to the C++ version. The difference is due to the different implementation languages as well as to some simplifications made during the conversion to java. For example, we used a simpler but slower version of a priority queue when converting to java. All timing is in arbitrary units comparing two algorithms run on the same single CPU computer. In practice even the worst non-AF distance calculation took an order of seconds on a 2 GHz PC.

Distances are calculated between the fixed partition $A(r, 1)$ of r ungrouped individuals and a partition B of r randomly grouped individuals. To allow for the direct comparison of a pair of algorithms, the partition-distance is calculated for exactly the same B partitions by the AF algorithm and the java version of the assignment

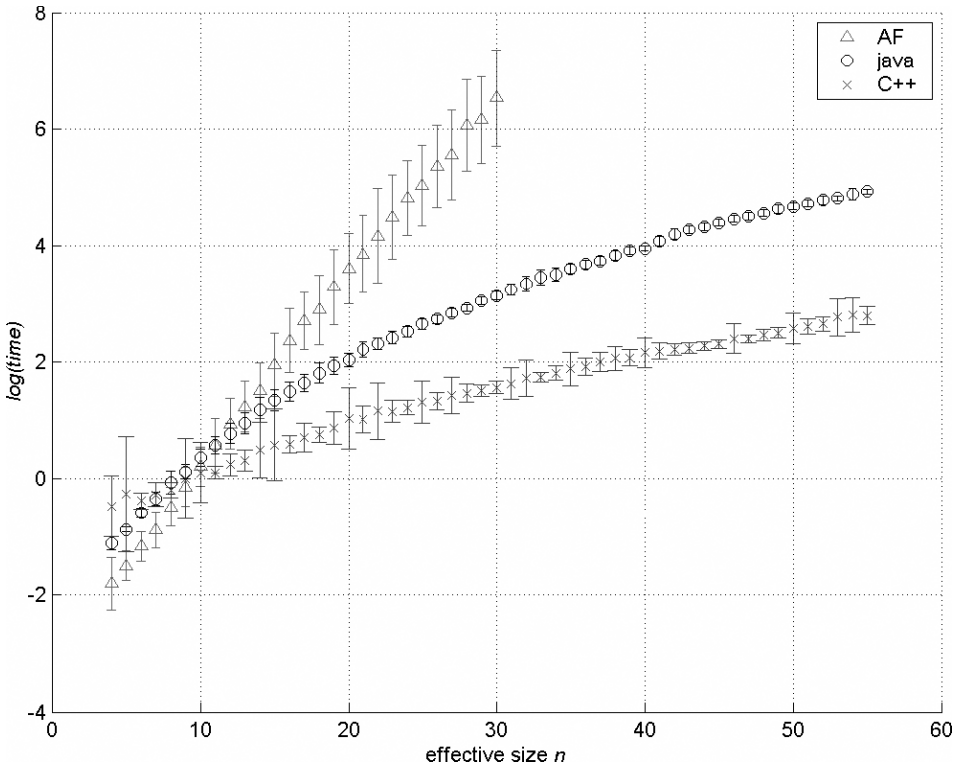

Fig. 1. Performance comparison of time taken to calculate partition-distance on the simulation set R1. The results for the assignment algorithm backed by the LEDA's C++ version of the weighted assignment method is denoted by crosses, while the java version is denoted by circles. The AF algorithm is denoted by triangles. Natural logarithm of time is displayed in arbitrary units against the effective partition size n. Each point is an average over 100 different distance calculations with the error bars showing ± 1 SD for that point.

algorithm for the smaller values of the effective sizes. For the higher values (above $n = 30$), the same pairwise comparison procedure was performed for the C++ and java versions (the C++ results are nevertheless displayed for the smaller sizes for visual continuity).

Figure 1 shows that the AF algorithm is in fact much slower than this paper's algorithm for all but very small values of effective size. The AF's performance is approximated to be $O(n^5)$ on the shown interval. It was stated by Gusfield (2002), not proven in the paper, that the AF algorithm is exponential-time in worst-cases as a function of the population size p. As expected the C++ backed assignment algorithm achieves $O(n^2)$ in line with the worst-case $O(n[n^2 + n\log(n)])$ reported by Mehlhorn and Näher (1999).

Simulation test R10

The second simulation test is arranged to emulate possible outcomes of accuracy testing in Butler *et al.* (2004) for a number of groups with each group containing 10 individuals. Partition A was created to be $A(r, 10) = \{a_1, \ldots, a_r\}$ such that each cluster a_i contained 10 elements giving the total population size $p = r \times 10$. Partition B was randomly created as described in the first simulation test R1. Distances are calculated between the fixed partition $A(r, 10)$ of r clusters of 10 individuals and a partition B of $p = r \times 10$ randomly grouped individuals. For example, five groups of 10 individuals yield the total population size of $p = 50$ which could randomly produce a B partition of only nine clusters, thus giving the effective size of $n = 9$.

Figure 2 further confirms the expected performance behavior of the paper's algorithm where the effective size n, not the population size p, contributes to the increase in time. For example, for an effective size $n = 20$ the population size could be as high as $p = 200$. On the other hand the performance of the AF algorithm is directly linked to the population size p showing a more dramatic increase in time, about $O(n^{10})$ over the displayed interval.

Simulation test RM

The third simulation test is arranged to emulate possible outcomes of accuracy testing for a number of groups with each group containing m individuals (excluding singletons). Partition A was created to be $A(r, m) = \{a_1, \ldots, a_r\}$ such that each cluster a_i contained m elements giving the total population size $p = r \times m$. Partition B was created by randomly moving each of the x individuals to a different group simulating the misclassification errors of a reconstruction algorithm.

Figure 3 illustrates the misclassification effect of x individuals for the family structures $A(5, 10)$, $A(5, 40)$ and $A(20, 10)$ studied by Butler *et al.* (2004) with $A(10, 10)$ added for continuity. Figure 3 demonstrates that the performance of this paper's algorithm does not significantly deteriorate over the considered range of x-moves since it depends mainly on the effective partition size n which has an upper bound of the corresponding r values in this simulation test.

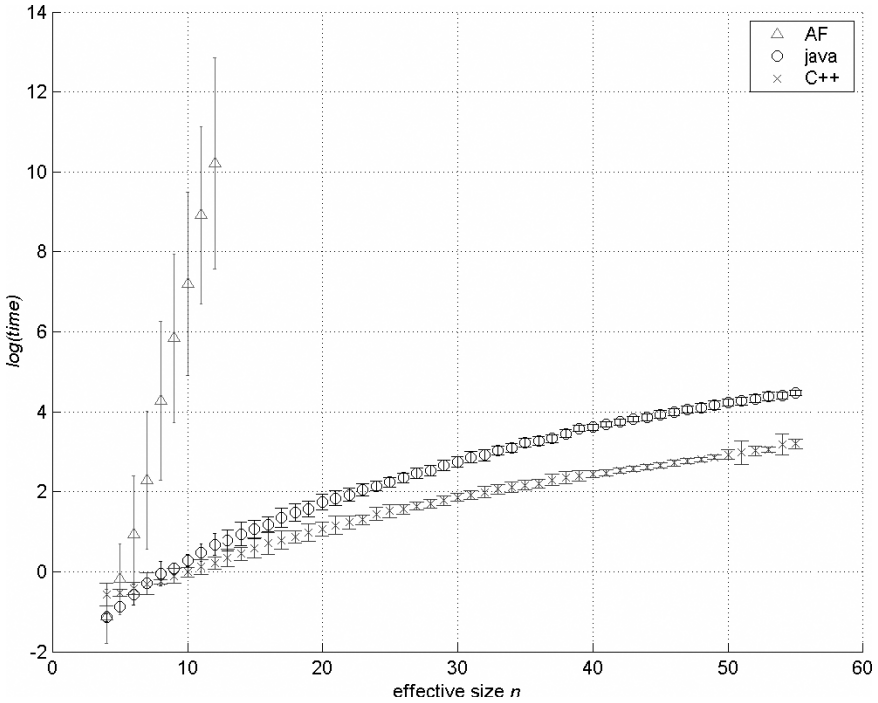

Fig. 2. The same as in Figure 1 but for the simulation set R10 and the higher values were taken to be above $n = 13$ for the pairwise comparison between the C++ and java versions.

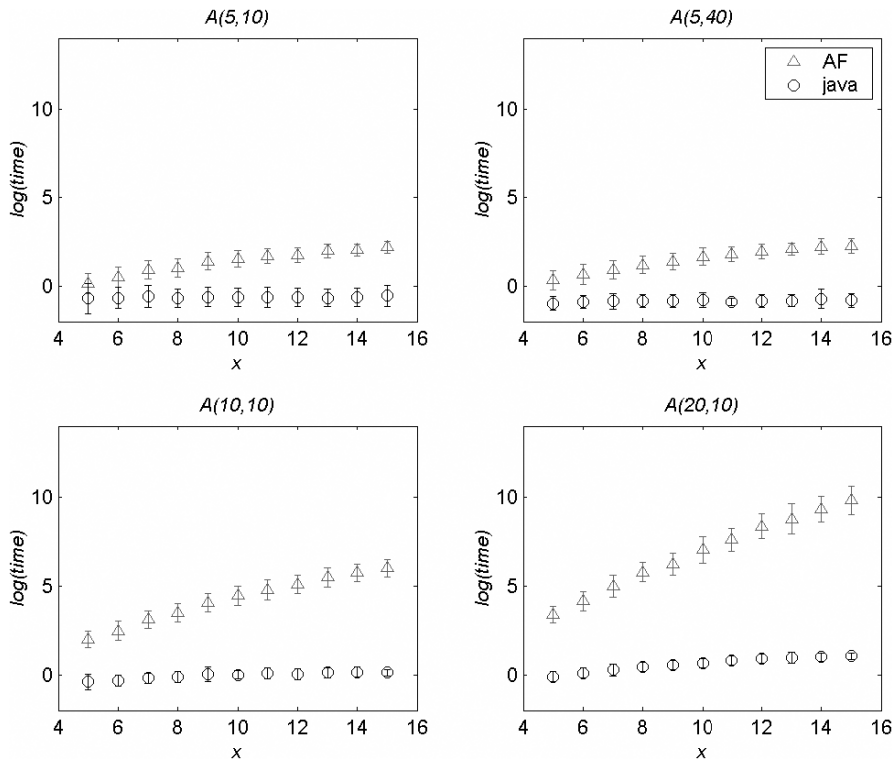

Fig. 3. The same as in Figure 1 but for the simulation set RM. Natural logarithm of time is displayed in arbitrary units against the number of randomly moved individuals x.

CONCLUSION

We achieved a reduction [using an approach different to that of Gusfield (2002)] of the partition-distance problem to the assignment problem, which has running time bounded above by $O(n^3)$, where n is the partition size. Further investigation into faster assignment problem algorithms could be undertaken. For example, Gabow and Tarjan (1989) have described an algorithm whose running time is bounded above by $n^{5/2}\log(nN)$, where N is the maximum absolute value of any edge cost. Note that in our application N is bounded above by the population size, so that $\log(N)$ is not a significant factor. It may also be the case that in practice, for biologically significant instances of the partition problem, the running time of the LEDA implementation of the assignment problem is substantially faster than the $O(n^3)$ upper bound. Our simulation results suggest this, and more work could be done to clarify this point.

ACKNOWLEDGEMENTS

We would like to thank Ross Crozier, Dan Gusfield, Christophe Herbinger and two anonymous reviewers for some helpful comments and suggestions.

REFERENCES

Almudevar,A. and Field,C. (1999) Estimation of single-generation sibling relationships based on DNA markers. *J. Agricult. Biol. Environ. Statist.*, **4**, 136–165.

Beyer,J. and May,B. (2003) A graph-theoretic approach to the partition of individuals into full-sib families. *Mol. Ecol.*, **12**, 2243–2250.

Butler,K. *et al*. (2004) Accuracy, efficiency and robustness of four algorithms allowing full sibship reconstruction from DNA marker data. *Mol. Ecol.*, **13**, 1589–1600.

Gabow,H.N. and Tarjan,R.E. (1989) Faster scaling algorithms for network problems. *SIAM J. Comput.*, **18**, 1013–1036.

Goodnight,K.F. and Queller,D.C. (1999) Computer software for performing likelihood tests of pedigree relationship using genetic markers. *Mol. Ecol.*, **8**, 1231–1234.

Gusfield,D. (2002) Partition-distance: a problem and class of perfect graphs arising in clustering. *Inform. Process. Lett.*, **82**, 159–164.

Jones,A.G. and Arden,W.R. (2003) Methods of parentage analysis in natural populations. *Mol. Ecol.*, **12**, 2511–2523.

Konovalov,D.A. *et al*.(2004) KinGroup: a program for pedigree relationship reconstruction and kin group assignments using genetic markers. *Mol. Ecol. Notes*, **4**, 779–782.

Luikart,G. and England,P.R. (1999) Statistical analysis of microsatellite DNA data. *Trend. Ecol. Evol.*, **14**, 253–256.

Mehlhorn,K. and Näher,S. (1999) *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge.

Papadimitriou,C.H. and Steiglitz,K. (1998) *Combinatorial Optimization: Algorithms and Complexity*, unabridged edn. Dover Publications.