BIOINFORMATICS APPLICATIONS NOTE Vol. 23 no. 9 2007, pages 1156–1158
doi:10.1093/bioinformatics/btm077

Genome analysis

rh_tsp_map 3.0: end-to-end radiation hybrid mapping with improved speed and quality control

Alejandro A. Schäffer[1], Edward Stallknecht Rice[1,2], William Cook[3] and Richa Agarwala[1,*]

[1]National Center for Biotechnology Information, National Institutes of Health, Department of Health and Human Services, Bldg. 38A, Room 10–03N, 8600 Rockville Pike, Bethesda, MD 20894, [2]Thomas Jefferson High School for Science and Technology, 6560 Braddock Road, Alexandria, VA 22312 and [3]Industrial and Systems Engineering, Georgia Tech, 765 Ferst Drive, Atlanta, GA 30332, USA

Received on January 19, 2007; revised on February 26, 2007; accepted on February 26, 2007

Advance Access publication March 1, 2007

Associate Editor: Alex Bateman

ABSTRACT

Summary: rh_tsp_map is a software package for computing radiation hybrid (RH) maps and for integrating physical and genetic maps. It solves the central mapping instances by reducing them to the traveling salesman problem (TSP) and using a modification of the CONCORDE package to solve the TSP instances. We present some of the features added between the initial rh_tsp_map version 1.0 and the current version 3.0, emphasizing the automation of many steps and addition of various checks designed to find problems with the input data. Iterations of improved input data followed by fast re-computation of the maps improves the quality of the final maps.

Availability: rh_tsp_map source code and documentation including a tutorial is available at ftp://ftp.ncbi.nih.gov/pub/agarwala/rhmapping/rh_tsp_map.tar.gz. CONCORDE modified for RH mapping is available in the directory http://www.isye.gatech.edu/~wcook/rh/. The QSopt library needed for CONCORDE is available at http://www2.isye.gatech.edu/~wcook/qsopt/downloads/downloads.htm

Contact: richa@helix.nih.gov, FAX: 301-480-2288. (Please send email concurrently with any fax.)

1 INTRODUCTION

Radiation hybrid (RH) mapping projects produce physical maps of sequence-tagged site markers on chromosomes and genomes. The input to the computational part of RH mapping is an $M \times P$ matrix of $\{0, 1, 2\}$, where M is the number of markers and P is the size of the 'panel' of clones. Each row is called an 'RHvector' for that marker. For each marker m and each clone c, a value of 0 means m is absent in c, 1 means m is present in c and 2 means the experiment gave ambiguous results. M may range from hundreds to thousands for entire genomes and up to low hundreds for single chromosomes. P is usually between 90 and 94, due to the prevalence of 96-well plates (allowing a few control wells), although theoretical considerations strongly suggest that to get a reliable map, P should increase as M increases (Ben-Dor and Chor, 1997). The basic principle of RH map computation is that markers with similar RHvectors should be close together in the map. The theory of RH map computation (see e.g. Lange *et al*., 1995) has been worked out under the assumption that the RHvectors contain all 0's and 1's, but this is hardly true in practice. For example, in the data set described in Murphy *et al*., (2007), only 0.68% of matrix entries are 2, but 39.82% of RHvectors contain at least one 2.

The initial publication describing rh_tsp_map 1.0 (Agarwala *et al*., 2000), a software package for constructing RH maps and integrating physical and genetic maps, introduced explicit consideration of matrix entries of 2. Versions of rh_tsp_map have been used to compute whole-genome maps of human (Agarwala *et al*., 2000), dog (Guyon *et al*., 2003), cow (Everts-van der Wind *et al*., 2005), a fish called the sea bream (Senger *et al*., 2006), rhesus macaque (Murphy *et al*., 2006) and cat (Murphy *et al*., 2007) as well as chromosome maps of other vertebrates such as horse (Goh *et al*., 2007). In this note, we describe some of the most important features available in the current version 3.0 of rh_tsp_map that have been added since version 1.0. These features were driven by the practical experience gained when computing RH maps and point to the basic problems with RH mapping—the biology for making hybrids using radiation and the lab protocols provide minimal checks on the correctness of the RHvectors. This contrasts with genetic mapping with microsatellites, in which the combination of the rules of inheritance and the polymorphism of the markers provides a much stronger check on the correctness and consistency of the allele calls. However, RH mapping has some major advantages over genetic mapping, for example, markers need not be polymorphic, scoring markers is easier, and the DNA can be kept in cell lines eliminating the costs of animal care and breeding. Due to the complementary nature of the two techniques, an integrated RH and genetic map is likely to be the most informative map for a genome.

Two principal contributions of rh_tsp_map 1.0 were:

(1) An implementation of a known reduction from the marker-ordering problem to the traveling salesman problem (TSP), for either the maximum likelihood estimate (MLE) or minimum obligate chromosome breaks (OCB) criterion, and use of the linkern program

*To whom correspondence should be addressed.

of CONCORDE (Applegate *et al*., 2006) to solve the TSP instances likely to optimality.

(2) Redefinition of the basic problem as selection of a set of markers that can be reliably ordered on a framework map. Ordering all markers is impractical at least because some markers are too close to have their order determined with current panel sizes, and because the mathematically optimal order of some markers depends on how the ambiguous (2) entries are interpreted in the formulation of either the MLE or OCB criterion.

In contrast, other RH mapping packages such as RHMAP (Boehnke *et al*.,1991), RHMAPPER (Slonim *et al*., 1997) and MultiMap (Matise and Chakravarti, 1995) try to order all the markers, and use heuristics that are unlikely to produce optimal maps. An independent test of an early version of `rh_tsp_map` showed that it is much faster and more robust than the incremental approach used by MultiMap and RHMAPPER (Hitte *et al*., 2003). As described in the Methods section below, recent changes improve both the speed and robustness of `rh_tsp_map`.

Finally, we now provide a tutorial with examples to guide users at each step of `rh_tsp_map`.

2 METHODS

With `rh_tsp_map` 3.0, the major steps to compute a genome-wide set of RH maps include:

(1) Identifying linkage groups of markers that belong together,

(2) Making a framework map for each linkage group,

(3) Testing the robustness of each framework map and dropping or improving the RHvectors for markers as needed,

(4) Placing additional markers with respect to each framework map and

(5) Making final maps, including the options to assign bins to markers that cannot be placed reliably and combining multiple linkage groups on a chromosome.

Data for all markers are available before step 1; the markers considered for placement at step 4 are those that do not make it onto the more reliably ordered framework maps that are constructed and tested in steps 2 and 3. Version 1.0 implemented only step 2. We have always allowed the alternative: to order all markers in a group. We now mention some of the modules for each step and highlight features specific to `rh_tsp_map`.

2.1 Identifying linkage groups

The key modules are `pairlods_dists` and `make_groups`, which compute pairwise logorithm of odds (LOD) scores and make linkage groups by single linkage clustering using marker pairs with LOD scores above a user-specified threshold. One novel feature is a quality control program, `discrepant_pairs`, that takes as input external evidence (e.g. from other maps, synteny or fluorescence *in situ* hybridization (FISH)) and finds marker pairs whose linkage contradicts the chromosome assignments suggested by the external evidence.

2.2 Making framework maps

An important change is that the software now solves the TSP instances to guaranteed optimality rather than likely optimality by using the `concorde` module of CONCORDE (Applegate *et al*., 2006) rather than `linkern`. Furthermore, in this application, `concorde` is orders of magnitude faster than `linkern` with the conservative parameters we recommend for `rh_tsp_map`. For example, computing framework maps with the three MLE criteria on the data set described in Murphy *et al*. (2007) using `concorde` takes under 13 s while using `linkern` takes over 3 h 20 min, although `linkern` also found optimal solutions. The cat framework maps contain a total of 1252 markers partitioned in 20 linkage groups.

To compare the running time of `concorde` to `linkern` in a case of a large number of markers in a linkage group, we considered all 1252 markers as one group, and randomly picked subsets of 626 markers and 313 markers. For these three sets of markers and the three MLE criteria, using `concorde` takes 1 min 38 s, 30 s and 9 s, respectively, while using `linkern` takes 1 h 8 min, 1 h 15 min and 1 h 10 min, respectively. Partitioning markers into 20 linkage groups took less time with `concorde` compared to having all the markers in one group because each individual problem was easier for `concorde`. However, the same behavior is not seen with `linkern` because the parameters specified find a solution from 10 starting points (called *restarts*) for each linkage group (for a total of 600 restarts with 3 criteria and 20 groups) compared to only a total of 30 restarts when all markers were in one group. As before, `linkern` also found an optimal solution for the set with 313 markers. However, `linkern` did not always find an optimal solution for the set with 626 and 1252 markers, suggesting that the parameters for `linkern` are not conservative enough for large problem sizes.

Because of decades of research into better algorithms and provable upper bounds for the length of TSP tours, instances with thousands of cities can be solved to guaranteed optimality quickly using `concorde`, even though TSP is NP-complete (Applegate *et al*., 2006). The `linkern` module was used in the original `rh_tsp_map` primarily because the `concorde` module could only be run together with commercial linear programming software at the time; once this limitation was overcome, some additional software integration was completed to put the RH distance functions into the `concorde` code.

Another recent addition allows users to express preferences for which marker to use in a framework map when two markers are too close to each other and, at most, one of them can be selected. This makes it possible to systematically increase the number of markers on the framework map about which one also has external evidence, improving the utility of the resulting framework maps for comparative genomics.

2.3 Testing framework maps

We added modules `map_eval` and `flips` for this step. `map_eval` removes one marker at a time from the input map and tests how much better its best position is to its second best position in the depleted map. `flips` tests the input map against all alternatives that locally permute any set of up to nine consecutive markers; this functionality is also available in $\mathrm{CAR}_{\mathrm{H}}^{\mathrm{T}}$AGENE (de Givry *et al*., 2005).

2.4 Placing more markers

Using the new module `place_interpolate`, each marker not on the framework map can be placed in its best framework map interval. The MLE or OCB score of the best placement is compared to the score of the second best placement, and only markers with a score difference above some user-specified threshold are retained on the map. To produce a globally integrated map combining framework markers and placed markers, we define and solve instances of a restricted TSP in which the framework markers are required to stay in the framework map order, placed markers are required to stay in their best intervals, but if there are multiple placed markers in the same framework interval, their order is chosen optimally. Order for placed markers is tested using

flips in a restricted mode that considers only those permutations that retain placed markers in the same interval. This global positioning of the placed markers avoids the incremental addition of markers used by other packages that causes the resulting maps to be highly sensitive to the order in which placed markers are added (Hitte *et al*., 2003).

2.5 Making final maps

We include new code to systematically handle markers with identical RHvectors, so that although only one of them is used in any flips test, all identical markers can appear in the final map with identical position. We include new options to assign markers that cannot be placed with high enough score to bins that are wider than intervals bounded by adjacent framework markers. This allows users to increase the number of markers shown with some positional information without corrupting the robustness of the total ordering of framework and placed markers.

3 IMPLEMENTATION

rh_tsp_map is implemented as a set of C programs. Some of these programs create UNIX shell scripts. For example, the frame_markers program creates a separate concorde_script for each distance criterion used. The CONCORDE package is implemented in C and available as source code. The concorde program solves TSP instances to optimality using the QSopt library, which is available as a file for numerous versions of UNIX.

ACKNOWLEDGEMENTS

This research was supported by the Intramural Research Program of the National Institutes of Health, National Library of Medicine and Office of Naval Research Grant N00014-03-1-0040. Thanks to William Murphy, Bhanu Chowdhary, Terje Raudsepp and Elisabete Amaral for giving us practical experience in constructing RH maps, which led to many of the improvements in rh_tsp_map. Funding to pay the Open Access charges was provided by the Intramural Research Program of the NIH, National Library of Medicine.

Conflict of Interest: none declared.

REFERENCES

Agarwala,R. *et al*. (2000) A fast and scalable radiation hybrid map construction and integration strategy. *Genome Res*., **10**, 350–364.

Applegate,D. *et al*. (2006) *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, New Jersey, USA.

Ben-Dor,A. and Chor,B. (1997) On constructing radiation hybrid maps. *J. Comp. Biol*., **4**, 517–533.

Boehnke,M. *et al*. (1991) Statistical methods for multipoint radiation hybrid mapping. *Am. J. Hum. Genet*., **49**, 1174–1188.

de Givry,S. *et al*. (2005) CAR$^{T}_{H}$AGENE: multipopulation integrated genetic and radiation hybrid mapping. *Bioinformatics*, **21**, 1703–1704.

Everts-van der Wind,A. *et al*. (2005) A high-resolution whole-genome cattle-human comparative map reveals details of mammalian chromosome evolution. *Proc. Natl Acad. Sci. USA*, **102**, 18526–18531.

Goh,G. *et al*. (2007) High-resolution gene maps of horse chromosomes 14 and 21: additional insights into evolution and rearrangements of HSA5 homologs in mammals. *Genomics*, **89**, 89–112.

Guyon,R. *et al*. (2003) A 1-Mb resolution radiation hybrid map of the canine genome. *Proc. Natl Acad. Sci. USA*, **100**, 5296–5301.

Hitte,C. *et al*. (2003) Comparison of MultiMap and TSP/CONCORDE for constructing radiation hybrid maps. *J. Hered*., **94**, 9–13.

Lange,K. *et al*. (1995) Statistical methods for polyploid radiation hybrid mapping. *Genome Res*., **5**, 136–150.

Matise,T.C. and Chakravarti,A. (1995) Automated construction of radiation hybrid maps using MultiMap. *Am. J. Hum. Genet*., **57**, A15.

Murphy,W.J. *et al*. (2005) A rhesus macaque radiation hybrid map and comparative analysis with the human genome. *Genomics*, **86**, 383–395.

Murphy,W.J. *et al*. (2007) A 1.5 megabase resolution radiation hybrid map of the cat genome and comparative analysis with the canine and human genomes. *Genomics*, **89**, 189–196.

Senger,F. *et al*. (2006) The first radiation hybrid map of a perch-like fish: the gilthead seabream (*Sparus aurata* L). *Genomics*, **87**, 793–800.

Slonim,D. *et al*. (1997) Building human genome maps with radiation hybrids. *J. Comp. Biol*., **4**, 487–504.