Journal of Biomedical Informatics 45 (2012) 931–937

Contents lists available at SciVerse ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Multiple ant colony algorithm method for selecting tag SNPs

Bo Liao *, Xiong Li, Wen Zhu, Renfa Li, Shulin Wang

The College of Information Science and Engineering, Hunan University, Changsha, Hunan 410082, China

ARTICLE INFO

Article history:
Received 5 June 2011
Accepted 19 March 2012
Available online 28 March 2012

Keywords:
Single nucleotide polymorphisms
Tag SNPs selection problem
Ant colony algorithm

ABSTRACT

The search for the association between complex disease and single nucleotide polymorphisms (SNPs) or haplotypes has recently received great attention. Finding a set of tag SNPs for haplotyping in a great number of samples is an important step to reduce cost for association study. Therefore, it is essential to select tag SNPs with more efficient algorithms. In this paper, we model problem of selection tag SNPs by MINIMUM TEST SET and use multiple ant colony algorithm (MACA) to search a smaller set of tag SNPs for haplotyping. The various experimental results on various datasets show that the running time of our method is less than GTagger and MLR. And MACA can find the most representative SNPs for haplotyping, so that MACA is more stable and the number of tag SNPs is also smaller than other evolutionary methods (like GTagger and NSGA-II). Our software is available upon request to the corresponding author.

1. Introduction

The search for the association between complex diseases and haplotype is the most interesting topic in the field of medicine or disease control and prevention. Several studies have proved that association studies using haplotype information generally outperform those using single SNP analyses [1]. Objective of these studies is to discover the relationship between genetic variations and such traits, by comparing genetic sequence and phenotypes of individuals sampled from a population. Although all single nucleotide polymorphisms (SNPs) can be used for indirect association studies to detect disease-related genetic variants, the complete screening of a gene or a chromosomal region is nevertheless an expensive undertaking. A key strategy to improve the efficiency of association studies is to select a subset of informative SNPs, called tag SNPs, for analysis [2]. Therefore, it is essential to use a small subset of informative SNPs accurately identifying haplotypes in a block.

The selection procedure is referred as haplotype tagging, which is a key process to save the cost for Genome Wide Association Study. The tag SNPs selection strongly depends on how the chosen SNPs will be used, and different sets of tag SNPs should be selected for fulfilling requirements of various genotyping platforms and projects [3]. For example, Carlson et al. [4] select maximally informative SNPs for association analysis and Chapman et al. [5] use haplotype tags to detect disease associations.

Tag SNPs selection is valuable, but it is proved to be a NP-hard problem [6], and computational science, which includes computational intelligence (CI), has recently become an important method for these complicate problems [7]. Many algorithms of tag SNPs selection have been developed in the past few years. Tag SNPs selection can follow two different strategies: the block-based and the block-free strategy. Block-based methods were based on the haplotype block structure of the human genome. The rationale is that the human genome can be partitioned into discrete blocks [8], and most members of a population share a very small subset of common haplotypes within each block. Since the number of distinct combinations of alleles (one of two or more alternative forms of a gene at corresponding loci on homologous chromosomes) within a block is relatively small [9], thus, selecting a small subset of SNPs that efficiently represent other SNPs in a given block is an important problem for reducing genotyping costs without losing the ability to detect disease associations. There are numerous block-based methods that include exact (e.g., [1]), approximation (e.g., [10]), and evolutionary (e.g., [11]) algorithms have been proposed to solve this problem. In a block-free method, tag SNPs are regarded as a subset of all SNPs, from which the remaining SNPs can be reconstructed. Block-free methods (e.g., [12,13]) do not need prior block partition or limit the diversity of haplotypes.

To select smaller tag SNPs and cost less time, a genetic algorithm, called GTagger (Genetic Tagger) [11], for the haplotype tagging SNPs (htSNPs) selection problem is designed. It is intended to find the smallest htSNPs set in blocks with relatively large number of SNP sites. However, GTagger cannot find the most representative SNPs for haplotyping, so that it is not stable and the number of tag SNPs is not small enough. He and Zelikovsky have introduced two novel approaches for informative SNP prediction based on multiple linear regression (MLR) [12] and support vector machines (SVMs). When the number of tags is increased to 30, MLR needs nearly half an hour to build a predictor. MLR takes a lot of time to completely reconstruct the predictor because of selecting a new tag. With

* Corresponding author. Fax: +86 731 88821417.
E-mail address: dragonbw@163.com (B. Liao).

1532-0464/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jbi.2012.03.003

more tags being selected, the running time increased sharply, so that MLR is not cost-effective enough.

In this study, we take the block-based approach and ant colony algorithm is continuously used several times to search for better solution, which we refer to as a multiple ant colony algorithm (MACA). The ant colony algorithm inspired by the observation of real ant colonies was first proposed by Dorigo and his colleagues [14] as a multi-agent approach to difficult combinatorial optimization problems. One of the important parameters in our method is the heuristic value (η). We design heuristic function with three heuristic factors (coverage, repeatability, margins). Larger granularity can save running time, and smaller granularity can select a smaller set of tag SNPs. For trade-off between running time and the number of tag SNPs, we use three sizes of granularity to build vertex (a set of SNPs). In our study, MACA is compared with two other latest evolutionary algorithms (GTagger and NSGA-II [3]) on the number of tags. The results show that MACA is more stable and the number of tag SNPs is smaller than others, as MACA can find the most representative SNPs for haplotyping. And, extensive experiments have shown that the running time of the MACA is also less than GTagger and MLR.

2. Problem formulation

The tag SNPs problem is regarded as being equivalent to the minimum test set problem from the beginning by Zhang et al. To better explain some concepts involved in our method, we will give a brief introduction to it in this section. Since we focus on biallelic SNPs, each haplotype is to be represented by a binary string set. The length of each haplotype is m and we denote it as $h_i = \{s_1, s_2, \ldots, s_m\}$, $s_i \in \{0,1\}$. Given a set of haplotypes $H = \{h_1, h_2, \ldots, h_m\}$ belonging to an arbitrary population, the purpose of this study is to find a smaller set of tag SNPs $T = \{t_1, t_2, \ldots, t_k\}$ (where k represents the selected number of tag SNPs) to recognize any proportion (even all) of haplotypes in H.

For the sake of convenience and without losing generality, we assume that the first haplotype is $h_1 = \{0,0,\ldots,0\}$, and if the SNP in the same column j in h_i ($i \neq 1$) is the same as h_1, then we let $h_{ij} = 0$, otherwise $h_{ij} = 1$. For example

$$H = \begin{bmatrix} h1 = [ATTT] \\ h2 = [GCCC] \\ h3 = [ATCT] \\ h4 = [GCTT] \end{bmatrix} \text{ is transformed to } H = \begin{bmatrix} h1 = [0000] \\ h2 = [1111] \\ h3 = [0010] \\ h4 = [1100] \end{bmatrix}$$

In this example, SNP2 and SNP3 are sufficient to identify each of the four haplotypes.

The set covering problem is a classical question in computer science and complexity theory. Given a $m * n$ matrix $A = [a_{ij}]$ with every element being 0 or 1, and $a_{ij} = 1$ represents that the jth column covers the ith row. Every column in Matrix A has a cost b_j. This problem is to find a subset with minimum total cost to cover every row. Use J to represent a subset of all columns, and y_j is a boolean variable. If $j \in J$, let $y_j = 1$ otherwise $y_j = 0$. This should be formulation for set cover problem,

$$\min f(y) = \sum_{j=1}^{n} b_j \times y_j \tag{1}$$

constrained by:

$$\sum_{j=1}^{n} a_{ij} \times y_j \geqslant 1 \quad (i = 1, 2, \ldots, m) \tag{2}$$

$$y_j \in \{0, 1\} \tag{3}$$

For the haplotype tagging problem, we take the cost of each SNP as 1, so the (1) is transformed to (4):

$$\min f(y) = \sum_{j=1}^{n} 1 \times y_j \tag{4}$$

Let $C(s_i)$ represents the set of covered haplotypes, and $|C(s_i)|$ is *coverage*. Set cover model has an important property (5) and we will make full use of it to construct heuristic function.

$$C(S_i + S_j) \supseteq C(S_i) + C(S_j) \tag{5}$$

Namely, when a set of SNPs S_j is added to S_i, there are some new covered haplotypes not existing in $C(S_i)$ or $C(S_j)$, and we consider it as a *margin*. When a set of SNPs S_j is added to S_i, $C(S_i) \cap C(S_j) \neq \emptyset$ may happen. Then, we define $|C(S_i) \cap C(S_j)|$ as a *repeatability*.

3. Methods

In this section, we purpose a method for finding a small subset of tag SNPs which can accurately identify haplotypes in cases or controls for Genome Wide Association Study (GWAS). In order to better describe our method, we divide this section into three subsections. Section 3.1 outlines our approach to solving minimum test cover, and the corresponding algorithm is in Section 4. After that, we separately introduce two important components (pheromone and heuristic value) of the ant colony algorithm in Sections 3.2 and 3.3.

3.1. Multiple ant colony algorithm (MACA)

In the natural world, ants leave pheromone trails and choose a path according to the concentration of pheromone, and the pheromone density is higher when the path is shorter. Thus, this positive feedback eventually leads all the ants to follow a shorter path. The idea of the ant colony algorithm is to simulate real the ant's behavior. Ant colony algorithm was first proposed by Dorigo and his colleagues as a multi-agent approach to difficult combinatorial optimization problems like the traveling salesman problem (TSP) and the quadratic assignment problem (QAP) [14]. There is currently a lot of ongoing activity in the scientific community to extend or apply ant-based algorithms to solve many different discrete optimization problems.

In this study, firstly, we aggregate $t(2^t \approx m$, m is the number of haplotypes, and t is granularity of a vertex) SNPs to form vertexes, and t SNPs can be successive or randomly (the experiment shows that the difference between successive and random is not much). One SNP cannot be aggregated in different vertexes. Ideally, t SNPs can cover 2^t ($2^t \leqslant$ m) haplotypes. After the ant colony algorithm (ACA) has found the best combination of vertexes with granularity t, we shrink t to half and use ACA again to optimize the selected vertexes. Finally, we directly shrink t to 1 and the last optimization process is executed. Larger granularity accelerates the convergence, and smaller granularity refines the solution. For the trade-off between running time and the number of tags, we gradually use t, $t/2$ and 1 to be the size of granularity, so that we recursively run the ACA algorithm three times in total, and our method is named MACA.

3.2. Ant-decision and Pheromone-update

In the ant colony algorithm, a key factor that influences ants decision-making is pheromone. When a SNP is selected by more ants, more pheromone is accumulated on this SNP, so that the probability of it being tag is bigger. When the ant colony algorithm is applied to the set cover problem, pheromone should be conserved on vertex (SNPs or single SNP), not on the path.

Since pheromone expressed by τ_i is conserved on the vertex, the probability with which an ant k chooses the vertex i to be part of the solution is: