AntMap: Constructing Genetic Linkage Maps Using an Ant Colony Optimization Algorithm

Hiroyoshi Iwata* and Seishi Ninomiya
National Agricultural Research Center, 3-1-1 Kannondai, Tsukuba, Ibaraki 305-8666, Japan

High-throughput assay of molecular markers enables to utilize large amounts of markers in linkage mapping. To incorporate numerous markers into a linkage map, the development of a highly efficient method for linkage mapping is indispensable. When the number of loci is large, locus ordering is a major difficulty in linkage mapping, since the number of possible orders becomes very large. To address this problem, we developed a new algorithm for locus ordering and used it in a newly developed computer program called AntMap. The algorithm is based on ant colony optimization, which is a set of metaheuristic algorithms inspired by the cooperative behavior of real ants in finding the shortest path from their nest to a food source. Using this algorithm, AntMap seeks the linear order of loci that minimizes the sum of adjacent recombination fractions or that maximizes the log likelihood of locus order. Analyses based on simulated data sets indicated that our algorithm displayed a high efficiency level. The high performance of the algorithm enabled to save time and labor, and also to validate an estimated order by bootstrap tests. Our algorithm and AntMap should enable to construct high-throughput systems for linkage mapping. AntMap is available under a GNU general public license at http://cse.naro.affrc.go.jp/iwata/antmap/index.html. Source codes and executables of AntMap can be obtained there.

Key Words: ant colony optimization (ACO), linkage mapping, locus ordering, traveling salesman problem (TSP), metaheuristics, computer program, simulation analysis.

Introduction

Linkage maps have become an essential tool for plant and animal breeding, since they provide a good mechanism for identifying key genes involved in targeted traits. Quantitative trait locus analysis enables to locate genes even for complex quantitative traits on linkage maps. Information about the location of genes can be further exploited in a breeding program through various approaches such as marker-assisted selection, marker-assisted introgression, and map-based cloning of targeted genes. In recent years, assays of molecular markers have become highly efficient, and increasing numbers of markers are available in many species. Future technological innovations should accelerate this trend. To incorporate the large amount of molecular markers into a linkage map, the development of highly efficient methods for constructing linkage maps is indispensable.

When the number of loci increases, locus ordering becomes the main difficulty in linkage mapping, since the sheer number of possible orders of loci becomes very large. With $n$ linked loci, the number is $n! / 2$. When $n = 100$, the number reaches a value of $4.7 \times 10^{157}$. In such a case, it is very difficult to examine all possible orders to determine the optimum one. Even when $n = 30$, we must use an efficient search algorithm that enables to identify a practically good order within a practical time period.

The problem of searching for the best (i.e., optimum) locus order can be considered as a special case of the traveling salesman problem (TSP) (Liu 1998). In the TSP, given a set of cities and the distances between each pair of them, the objective is to identify a round trip with minimum total length in which each city is visited exactly once. In locus ordering, loci and distances between loci, e.g., the recombination fraction, can be considered to correspond to the cities and distances between cities, respectively, in the TSP. To solve the TSP, a number of efficient search algorithms, such as simulated annealing (Kirkpatrick et al. 1983), tabu search (Glover 1989) and genetic algorithms (Holland 1975), have been proposed. In recent years, in some methods for linkage mapping, heuristic algorithms for the TSP have been used to solve the problem of locus ordering (e.g. Schiex and Gaspin 1997, Jansen et al. 2001, Mester et al. 2003a, 2003b).

In the present paper, we introduced an algorithm for locus ordering based on ant colony optimization (ACO) (Dorigo et al. 1996), which is a set of algorithms inspired by the cooperative behavior of real ants in finding the shortest path from their nest to a food source. In recent years, ACO has been used successfully to solve various types of discrete optimization problems (Dorigo and Stützle 2004), and it is one of the best algorithms for the TSP (Dorigo et al. 1996).
To apply ACO to linkage mapping, we devised algorithms to handle the locus-ordering problem, to satisfy the conventional maximum likelihood criterion, and to be robust against missing marker-genotype data. We used our algorithm in a new computer program that we named AntMap. We conducted simulation studies to determine the performance of our algorithm. We describe here the details of our algorithm, the outline of the AntMap system, and the results of simulation studies. Finally, we examined several advantages of our system.

Algorithm

ACO is a metaheuristic algorithm in which a colony of artificial ants cooperates in finding a good solution to a discrete optimization problem such as the TSP. In the TSP, a given set of cities has to be traversed so that every city is visited exactly once and the tour ends in the initial city. The optimization goal is to identify the shortest possible route. ACO can be applied to the TSP with the following steps: (1) Each of m ants begins a tour from an arbitrary city. (2) An ant moves to a city with a probability that is a function of the distance and the amount of pheromone already accumulated on the trail until it completes a tour. (3) Every ant secretes trail pheromone on its route, which evaporates with time. (4) The optimum tour found after iterating steps 1 to 3 T times is considered to correspond to the solution. At iteration t, an ant k in city i chooses city j to move to with the following probability:

\[ p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^{\alpha} [d_{ij}]^{-\beta}}{\sum_{l \in N^k} [\tau_{il}(t)]^{\alpha} [d_{il}]^{-\beta}} \quad \forall j \in N^i, \]  

(1)

where \( \tau_{ij}(t) \) is the amount of pheromone on the trail between cities i and j at iteration t, \( d_{ij} \) is the distance between cities i and j, and \( N^i \) is the set of cities that ant k has not yet visited. Parameters \( \alpha \) and \( \beta \) determine the relative influence of the pheromone and proximity (i.e., inverse of distance), respectively. After completing a tour, each ant deposits the pheromone according to the following rule:

\[ \Delta \tau_{ij}^k(t) = \begin{cases} Q/L^k(t) & \text{if } (i,j) \in T^k(t) \\ 0 & \text{if } (i,j) \notin T^k(t) \end{cases}, \]  

(2)

where Q is a constant, \( L^k(t) \) is the length of the tour undertaken by the ant k at iteration t, and \( T^k(t) \) is the set of trails between cities constituting the tour. The amount of pheromone on each trail is updated as follows:

\[ \tau_{ij}(t+1) = \rho \tau_{ij}(t) + \sum_{k=1}^{m} \Delta \tau_{ij}^k(t), \]  

(3)

where \( \rho \) represents the persistence of the pheromone trails.

In our system, loci and the absolute values of log likelihood (or the recombination fraction) between loci were considered to correspond to the TSP cities and distances between cities, respectively. The locus-ordering problem, however, differed from the conventional TSP in that the path was not closed (i.e., the “final leg of the journey” was not required). To account for this condition, we modified the system as follows: (1) Ants secrete the pheromone also in the cities at the starting and end points of a tour. (2) Ants select the starting point not randomly but with a probability that is a function of the amount of pheromone. That is, at iteration t, an ant k starts from city i with the following probability:

\[ p_{ik}^k(t) = \frac{[\tau_{ik}(t)]^{\alpha} [d_{ik}]^{-\beta}}{\sum_{l \in N^k} [\tau_{il}(t)]^{\alpha} [d_{il}]^{-\beta}} \quad \forall j \in N^i, \]  

(4)

where \( \tau_{ik}(t) \) is the amount of pheromone at city i at iteration t. The rule for secreting the pheromone in a city is as follows:

\[ \Delta \tau_{ik}^k(t) = \begin{cases} Q/L^k(t) & \text{if } i \in S^k(t) \\ 0 & \text{if } i \notin S^k(t) \end{cases}, \]  

(5)

where \( S^k(t) \) is the set of starting and terminal cities of a tour. The amount of pheromone in each city is updated as follows:

\[ \tau_{ik}(t+1) = \rho \tau_{ik}(t) + \sum_{k=1}^{m} \Delta \tau_{ik}^k(t). \]  

(6)

When a recombination fraction between loci is considered to correspond to the distance between the TSP cities, our system seeks the order minimizing

\[ \text{SARF} = \sum_{i=1}^{N-1} \theta_{i,i+1}, \]  

(7)

where \( N \) is the number of loci, and \( \theta_{i,i+1} \) is the estimate of the recombination fraction between adjacent (i.e., ith and (i+1)th) loci in a given order. This criterion, called the sum of adjacent recombination fractions (SARF), is used as a criterion in locus ordering (Olson and Boeckne 1990, Falk 1992, Liu 1998, Mester et al. 2003b). When the absolute values of log likelihood (ALL) between loci are considered to correspond to the distance between the TSP cities, our system seeks the order minimizing

\[ \text{ALL} = \sum_{i=1}^{N-1} n_{i,i+1} \{ \theta_{i,i+1} \log \theta_{i,i+1} + (1 - \theta_{i,i+1}) \log (1 - \theta_{i,i+1}) \}, \]  

(8)

where \( n_{i,i+1} \) is the number of informative sample sizes for estimating the recombination of adjacent (i.e., ith and (i+1)th) loci in a given order. The log likelihood always takes a negative value because both \( \theta_{i,i+1} \) and \( (1 - \theta_{i,i+1}) \) are lower than 1. Thus, the minimization of Eq. 8 is synonymous with the maximization of

\[ \text{LL} = \sum_{i=1}^{N-1} n_{i,i+1} \{ \theta_{i,i+1} \log \theta_{i,i+1} + (1 - \theta_{i,i+1}) \log (1 - \theta_{i,i+1}) \}. \]  

(9)

This is the log likelihood (LL) for a locus order on the assumption that there is no interference, i.e., recombination events occur independently for each interval. That is, in this case, the algorithm seeks the optimum order based on the conventional maximum likelihood criterion.

An optimization algorithm searches for an optimum solution by iteratively transforming a current candidate solution into a new solution, and hopefully results in the identification of a global optimum. An algorithm, however, is
sometimes trapped by a poor local optimum in which the criterion value of a solution is much worse than that of a global optimum solution. In our preliminary studies, the system described above sometimes converged to a poor local optimum when some observations were missing. That is, an order estimated by the system sometimes showed a worse criterion value than a simulated (i.e., true) order. To enable our system to avoid such a situation, we introduced a mechanism of random and elite selections, as suggested by Nakamichi and Arita (2004). Random selection is a simple operation whereby a city is selected among unvisited cities with equal probability and thus enables the system to escape from a poor local optimum. The random selection rate $\lambda$ is the probability that random selection occurs each time an ant selects the next city or the starting point. That is, with the probability $\lambda$, an ant selects the next city or the starting point randomly regardless of the amount of pheromone, while otherwise (i.e., with the probability $1-\lambda$), the ant selects the next city with Eq. 1 or the starting point with Eq. 4. Elite selection, which is opposite to random selection, places extra emphasis on the best tour found so far, and accelerates the convergence to an optimum. That is, the elite selection covers the shortfall in the convergence force caused by the random selection. The elite selection can be achieved by the modification of the pheromone-updating rules. That is, to introduce a mechanism of elite selection, the pheromone-updating rules are modified as follows:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \sum_{k=1}^{n} \Delta \tau_{ij}(t) + \sigma \Delta \tau_{ij}^{*}(t),$$

(10)

$$\Delta \tau_{ij}^{*}(t) = \begin{cases} \frac{Q/L^{*}(t)}{f(i,j) \in T^{*}(t)} & \text{if } (i,j) \in T^{*}(t) \\ 0 & \text{if } (i,j) \notin T^{*}(t) \end{cases},$$

(11)

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \sum_{k=1}^{n} \Delta \tau_{ij}(t) + \sigma \Delta \tau_{ij}^{*}(t),$$

(12)

$$\Delta \tau_{ij}^{*}(t) = \begin{cases} \frac{Q/L^{*}(t)}{f(i,j) \in S^{*}(t)} & \text{if } i \in S^{*}(t) \\ 0 & \text{if } i \notin S^{*}(t) \end{cases},$$

(13)

where $\sigma$ is a weighting parameter, $L^{*}(t)$ is the length of the best tour identified at iteration $t$, $T^{*}(t)$ is the set of trails between cities that constitutes the best tour, and $S^{*}(t)$ is the set of the starting and terminal cities of the best tour. In our system, both random and elite selections are implemented in combination. As suggested by Nakamichi and Arita (2004), this system is expected to improve the performance of the ACO algorithm in the balance between diversification (i.e., exploration of the search space) due to random selection and intensification (i.e., exploitation of the previous solutions) due to elite selection.

In summary, the algorithm with the random and elite selections proceeds as follows: (1) Each of the $n$ non-elite ants selects the starting city randomly (with probability $\lambda$) or according to the probability described in Eq. 4 (with probability $1-\lambda$). (2) During a tour, each of the non-elite ants selects the next city randomly (with probability $\lambda$) or according to the probability described in Eq. 1 (with probability $1-\lambda$). (3) After completing a tour, every non-elite ant secretes pheromone on its route according to Eqs. 2 and 5. $\sigma$ elite ants also secrete pheromone on the route of the best tour according to Eqs. 11 and 13. The rules for updating the amount of pheromone are included in Eqs. 10 and 12. (4) After iterating steps 1 to 3 $T$ times, the shortest tour is considered to correspond to the solution (i.e., an estimated locus order). In the algorithm without the random and elite selections, $\lambda$ and $\sigma$ are set to zero.

### Computer program

AntMap consists of a graphical-user-interface (GUI)-based application for linkage mapping (Fig. 1). AntMap is written in Java and runs on various operating systems (e.g., Windows, Linux, Solaris, or Mac OS) under the Java 2 Platform Standard Edition (J2SE) Java Runtime Environment (JRE). That is, AntMap can be run on any system in which J2SE JRE has been installed.

AntMap enables to analyze data derived from progenies of several types of crosses, including $F_2$ intercrosses, $F_2$ backcrosses, recombinant inbred lines, and doubled haploid lines. The input file format of AntMap is identical to the *.raw files required by MAPMAKER (Lander et al. 1987). The current version of AntMap, however, does not support two types of crosses, viz., $F_3$ intercross by self-mating ($F_3$ self) and recombinant inbred lines by sib-mating (RI-sib), which are supported by MAPMAKER/EXP.

Before locus ordering, a user can perform a segregation test and linkage grouping. In linkage grouping, two grouping methods, i.e., “nearest neighboring locus” and “all combinations”, can be selected. In the former, a group is formed by sequentially combining a locus which shows the lowest recombination value against it. This algorithm is used in MAPL (Ukai et al. 1991). In the latter, similar results to those of the “group” command of MAPMAKER can be obtained (Lander et al. 1987).

In locus ordering, a user can choose two optimization criteria: LL (ALL) or SARF. As described above, AntMap will seek a locus order that maximizes LL (i.e., minimizes ALL) or minimizes SARF. The map distance between markers is calculated from the order estimated by using Haldane’s map function (Haldane 1919) or Kosambi’s map function (Kosambi 1944). Finally, the estimated linkage map is graphed as line art, which can be saved as an image file.

In AntMap, the reliability of the estimated order of loci can be examined by using a bootstrap test (Manly 1998), i.e., a method for estimating the sampling distribution of an estimator by resampling with replacement from the original sample. In a bootstrap test, a random sample of size $n$ is drawn with replacement from the original sample of size $n$, and estimates are obtained from the random sample. After this operation is repeated many times (e.g., 100–1000 times), the empirical distribution of estimates is obtained. In AntMap,
by resampling individuals with replacement from the original sample as proposed by Liu (1998), the probability that a locus is located at the estimated order is calculated. Finally, a user can obtain a linkage map whose reliability is indicated by bootstrap values (Fig. 2).

The amount of time required for locus ordering is small, but it depends on the ACO parameter settings (e.g., the number of iterations) as well as the data set size (e.g., the number of loci). For example, a data sample that contains 6 chromosomes and 112 loci can be ordered in about 22 s with the default settings of AntMap (Intel Pentium M 1.6 GHz).

**Simulation study**

*Simulated data set*

In our simulation studies, a random data set for $K$ individuals from an $F_2$ population and for $N$ markers located on a chromosome was generated as follows:

1. $N-2$ markers were randomly located on a chromosome 150 cM long. That is, each marker was located at $x$ cM by drawing $x$ from a uniform distribution between 0 and 150. We located the remaining two markers at both ends (0 and 150 cM) of the chromosome. We renumbered the markers from one end to the other (i.e., after renumbering, the $i$-th marker was located at the $i$-th position in the simulated order).

2. The mode of inheritance of each marker was set to dominance with probability $p$, and set to codominance with probability $1-p$. We assumed that all the dominant

Fig. 1. A typical screenshot of the computer program AntMap.

**Fig. 2.** An example of a linkage map with bootstrap values (in parentheses). The bootstrap values can be considered to correspond to the probability that a locus is located at its estimated position. From these values, a researcher can identify the markers whose estimated positions may not be reliable. For example, both C3M13 and C3M14 in this map showed a comparatively low bootstrap value (i.e., 47%), indicating that the positions of C3M13 and C3M14 are likely to be reverse. The bootstrap values also indicate that the positions of C3M1 and C3M2 are possibly reverse, but the probability of reversion was much lower than that of C3M13 and C3M14.
markers were linked in coupling-phase. Although the assumption may be unrealistic, we deliberately avoided the inclusion of repulsion-phase dominant markers because the recombination between repulsion-phase markers cannot be adequately estimated in an F2 generation (Ukai 2000).

3. 2K gametes were randomly created. For each gamete, we first randomly chose the allele for the first marker in the set (“A”, “B”) with a probability of 0.5. Then for each successive marker, the allele was flipped with a probability equal to the recombination fraction. The recombination fraction r was calculated from the map distance d (cM) to the successive marker with the equation $r = 0.5(1 - \exp(-0.02d))$ (Haldane 1919).

4. Each of K individuals was generated by combining two of the 2K gametes.

5. A marker phenotype of each individual was determined as follows. For dominant markers, the phenotype was set to “A+” when the genotype was “AA” or “AB”, or to “BB” when the genotype was “BB”. For codominant markers, the phenotype was set to be identical to the genotype. In the input file of AntMap, the marker phenotypes “AA”, “AB”, “BB” and “A−” were coded as ‘A’, ‘H’, ‘B’ and ‘D’, respectively.

6. To incorporate “missing data”, each of the marker phenotypes generated was deleted with probability and replaced by a missing value, viz. ‘-’.

Each simulated data set was built using this process.

**Simulation methods**

To determine the performance of our algorithm, we conducted two kinds of simulation studies. The parameters of ACO were set as shown in Table 1 unless otherwise specified. The criterion to be optimized was the absolute value of LL.

First, to examine the effect of the random and elite selections, we compared locus orders estimated by the algorithms with and without the selections (Simulation 1). The algorithm without the selections can be obtained by setting the ACO parameters for the occurrence of random selection (i.e., $\lambda$) and the weight for elite selection (i.e., $\alpha$) to zero (see Table 1). One thousand simulated data sets generated from the condition \{N = 100, K = 200, p = 0.2, and q = 0.05\} were solved by both algorithms. In this simulation, we ran the optimization operation twice for each data set by each algorithm in order to examine the stability of a solution. That is, we performed 4000 (1000 data sets $\times$ 2 algorithms $\times$ 2 replications) optimization operations in total.

Second, to examine the effect of the number of markers included, we conducted simulations by varying the number of markers from 100 to 1000 (Simulation 2). One thousand sets of simulation data were generated from each of the following conditions: \{N = 100, K = 200, p = 0.0, q = 0.0\}, \{N = 300, K = 600, p = 0.0, q = 0.0\}, \{N = 500, K = 1000, p = 0.0, q = 0.0\} and \{N = 1000, K = 2000, p = 0.0, q = 0.0\}. To prevent a rise in the number of markers in which recombination with adjacent markers did not occur, the number of individuals (K) was also set to a size proportional to the number of markers (N). That is, we performed 4000 (4 conditions $\times$ 1000 datasets) optimization operations in total.

To evaluate the accuracy of an estimated order in each optimization operation, we used the following coefficient:

$$C_q = \frac{1}{N} \sum_{i=1}^{K} \delta_i,$$

where $\delta_i$ takes a value of 1 when the marker at the i-th position is identical between estimated and true (i.e., simulated) orders and 0 otherwise. That is, the coefficient $C_q$ represents the proportion of the markers located at the true positions. To evaluate the quality of an estimated order, we also compared the values of the optimization criterion (i.e., the absolute value of LL) between estimated and true orders for each optimization operation, and obtained the proportion of the cases in which the value of an estimated order was lower than or equal to the value of the true order (Pq).

In Simulation 1, we ran the optimization operation twice to examine the stability of a solution. To quantify the stability, we used the following coefficient:

$$C_{rep} = \frac{1}{N} \sum_{i=1}^{K} e_i,$$

where $e_i$ takes a value of 1 when the marker at the i-th position is identical between the two orders estimated and 0 otherwise. That is, the coefficient $C_{rep}$ represents the proportion of the markers located at identical positions between the replicated operations. In Simulation 1, we also compared the values of the optimization criterion between the two algorithms. We compared the better (i.e., lower) value of the replicated optimization operations between the two algorithms, and calculated the proportion of the cases in which the value of one algorithm was lower than or equal to the other (Pq).

**Simulation results**

In Simulation 1, the coefficients $C_{rep}$ and $P_q$ showed higher values for the algorithm with the random and elite selections than for the algorithm without the selections (Table 2). The higher value of $C_{rep}$ indicated that the stability of a solution was improved by the selections, and the higher
value of $P_\alpha$ indicated that the capability for obtaining a better solution had increased owing to the selections. $P_\alpha$ in the algorithm with the selections reached a value of 0.993, indicating that the algorithm without the selections gave a better solution than the algorithm with the selections in only 7 out of 1000 cases. Moreover, the value of the optimization criterion of an estimated order was worse than that of the true order in 12 out of 2000 optimization operations (i.e., $P_\beta = 0.994$) in the algorithm without the selections, but it was always better than or equal to the value of the true order (i.e., $P_\beta = 1.000$) in the algorithm with the selections.

In Simulation 2, over 99% of the markers were correctly located at the true positions (i.e., $C_q > 0.99$ on average) in all the conditions (Table 3). Surprisingly, the value of $C_q$ increased as the number of markers to be ordered increased, indicating that variation in the number of markers in the range up to 1000 did not affect the quality of an estimated order. When $N = 1000$, although an estimated order displayed a worse criterion value than the true order in 99 out of 1000 cases (i.e., $P_\alpha = 0.901$), this did not lead to a serious deterioration of the estimated order ($C_q = 0.997$ on average in these 99 cases).

### Table 2. Results of simulation study with a comparison of the capability between algorithms with and without random and elite selections (Simulation 1)

<table>
<thead>
<tr>
<th>Algorithm with random and elite selections</th>
<th>Algorithm without random and elite selections</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_q^{1)}$</td>
<td>$C_q^{1)}$</td>
</tr>
<tr>
<td>(0.0271)</td>
<td>(0.0443)</td>
</tr>
<tr>
<td>$C_{\alpha}^{2)}$</td>
<td>$C_{\alpha}^{2)}$</td>
</tr>
<tr>
<td>0.9942</td>
<td>0.9686</td>
</tr>
<tr>
<td>(0.0147)</td>
<td>(0.0460)</td>
</tr>
<tr>
<td>$P_\alpha^{3)}$</td>
<td>$P_\alpha^{3)}$</td>
</tr>
<tr>
<td>1.000</td>
<td>0.994</td>
</tr>
<tr>
<td>$P_\beta^{4)}$</td>
<td>0.993</td>
</tr>
</tbody>
</table>

1) Mean and SD (in parentheses) of the proportion of markers located at the true positions.
2) Mean and SD (in parentheses) of the proportion of markers located at identical positions in replicated optimization operations.
3) Proportion of cases in which the criterion value of an estimated order was lower than or equal to that of the true order.
4) Proportion of cases in which the criterion value of one algorithm was lower than or equal to that of the other algorithm.

### Discussion

Our algorithm for locus ordering offered several advantages as follows: 1) simplicity and convenience for use in a computer program; 2) rapidity; and 3) interactive operations by humans in the ordering process were not required. The second and third advantages not only enabled to save time and labor, but also to validate an estimated order by using bootstrap tests. The third advantage ensured the objectivity of the estimation process, because the ordering process did not involve human intervention. Finally, it enabled to estimate a locus order with sufficient accuracy in practical applications, as suggested by the simulation studies.

In our algorithm for locus ordering, we used the random and elite selections proposed by Nakamichi and Arita (2004). The results of Simulation 1 suggested that the selections improved the performance of the algorithm. That is, in the case of the algorithm with the selections, a better (or at least equally good) solution was obtained than in the case of the algorithm without the selections in most cases (i.e., 99.3%), while the reverse case was comparatively rare (i.e., 14.0%). Moreover, the stability of a solution was also enhanced owing to the selections. As suggested by Nakamichi and Arita (2004), random selection increased the exploration ability of the system, helping the system to escape from local optima that trap the algorithm without selection.

Although the stability of a solution was considerably enhanced by the algorithm with the selections, an estimated order was not always congruent between replicated optimization operations. In Simulation 1, 169 out of 1000 cases showed some discrepancies between the replications (i.e., $C_{\alpha} < 1.0$), although the proportion of the markers showing a discrepancy was not large (i.e., less than 0.6% on average). One way to deal with the instability of a solution might be to confirm an estimated order by repeating the optimization operation. In AntMap, the optimization operation is repeated several (say three) times for each data set, and then the optimum solution among the repetitions is adopted as an estimated order.

To keep pace with future technological innovations of molecular assays, it might become necessary to map hundreds of markers in a single linkage group. The results of Simulation 2 suggested that increasing the number of markers did not affect the quality of an estimated order in the range up to 1000 markers. In contrast to our preliminary expectation, the quality of an estimated order, which was evaluated by $C_q$, increased as the number of markers increased. Thus, our algorithm could be efficiently used in the linkage mapping of hundreds of markers, and meet the needs of high-throughput linkage mapping in the future.

In the ACO algorithms, several control parameters (see Table 1) that influence the performance (i.e., the quality of solutions found and the time required to reach these solutions) of algorithms are being used. In our system, the values of the parameters were based on settings empirically recognized as "good ones" (Dorigo and Stützle 2004). Since
optimal values are usually problem-dependent, the tuning of
the values of the parameters may further improve the perform-
ance of our algorithm. The tuning, however, can be per-
formed through elaborate empirical analyses (Engelbrecht
2005). Although these parameters are usually hand-tuned or
optimized via cross-validation techniques, approaches to au-
tomating the setting of parameters have also been proposed
(Engelbrecht 2005).

Although only the absolute value of LL was chosen as
an optimization criterion in the Simulations 1 and 2, AntMap
also provides SARF as a criterion in locus ordering, because
SARF has been found to be the optimum criterion in the sim-
ulation study (Olson and Boehnke 1990) and used as a crite-
the difference between the two criteria, we analyzed simul-
data with the criteria. One thousand sets of simulation
data were generated from the condition \( N = 100, K = 200, \, p = 0.0, q = 0.0 \). As a result, we found that the quality of an
estimated order and the stability of a solution were slightly
higher in LL than in SARF (on average, \( C_p = 0.989 \) and \( C_{SARF} = 0.998 \) in SARF, while \( C_p = 0.995 \) and \( C_{SARF} = 1.000 \) in LL).
When we analyzed simulation data by varying the number of
dominant markers and the proportion of missing data, the
results showed the same tendency (data not shown). Thus, the
use of LL as an optimization criterion may be preferable
for locus ordering with our algorithm, although the difference
between the two criteria may not be large.

In the present study, we developed the novel computer
program AntMap, in which our algorithm was used to order
loci. The source code and executables of AntMap will be
useful in future practical applications of our algorithm.
For example, the computational part of AntMap can be easily in-
corporated into other linkage mapping programs, since it is
coded by an objective computer language. The source code
and executables of AntMap are available under a GNU gen-
eral public license at http://cse.naro.affrc.go.jp/iwatah/antmap/
index.html. A user’s manual and a brief tutorial can be ob-
tained there too.

Acknowledgements

This work was partially supported by the Ministry of
Education, Science, Sports and Culture with a Grant-in-Aid
for Young Scientists (B), No. 15780007, 2003–2004.

Literature Cited

Dorigo, M., V. Maniezzo and A. Colomi (1996) The ant system: opti-

London.

Intelligence. John Wiley & Sons, Chicester.


Haldane, J.B.S. (1919) The combination of linkage values and the
calculation of distances between the loci of linked factors. J.
Genet. 8: 299–309.

Holland, J. (1975) Adaptation in Natural and Artificial Systems. MIT
Press, Cambridge, MA, USA.


Kosambi, D.D. (1944) The estimation of map distances from recombi-

Lander, E.S., P. Green, J. Abrahamson, A. Barlow, M.J. Daly, S.E.
computer package of constructing primary genetic linkage

Analysis. CRCPress, New York.

Manly, B.F.J. (1998) Randomization, Bootstrap and Monte Carlo Meth-

Efficient multipoint mapping: making use of dominant reposition-

Constructing large-scale genetic maps using an evolutionary strat-

Nakamichi, Y. and T. Arita (2004) Diversity control in ant colony opti-

Olson, J.M. and M. Boehnke (1990) Monte Carlo comparison of pre-
liminary methods for ordering multiple genetic loci. Am. J.

Schiex, T. and C. Gaspin (1997) CARTHAGENE: construction and
joining maximum likelihood genetic maps. ISMB 5: 258–267.

Ukai, Y. (2000) Genetic Analysis at the Genomic Level: Map and
QTL University of Tokyo Press, Tokyo (in Japanese).

Ukai, Y., R. Ohsawa and A. Saito (1991) MAPL: a package of micro-
computer programs for RFLP linkage mapping. Rice Genet.