BMC Bioinformatics

BioMed Central

Methodology article

Open Access

Evolutionary algorithms for the selection of single nucleotide polymorphisms

Robert M Hubley[1], Eckart Zitzler[2] and Jared C Roach*[1]

Address: [1]Institute for Systems Biology, Seattle, WA, USA and [2]Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland

Email: Robert M Hubley - rhubley@systemsbiology.org; Eckart Zitzler - zitzler@tik.ee.ethz.ch; Jared C Roach* - jroach@systemsbiology.org

* Corresponding author

Published: 23 July 2003

BMC Bioinformatics 2003, **4**:30

Received: 07 March 2003

Accepted: 23 July 2003

This article is available from: http://www.biomedcentral.com/1471-2105/4/30

Abstract

Background: Large databases of single nucleotide polymorphisms (SNPs) are available for use in genomics studies. Typically, investigators must choose a subset of SNPs from these databases to employ in their studies. The choice of subset is influenced by many factors, including estimated or known reliability of the SNP, biochemical factors, intellectual property, cost, and effectiveness of the subset for mapping genes or identifying disease loci. We present an evolutionary algorithm for multiobjective SNP selection.

Results: We implemented a modified version of the Strength-Pareto Evolutionary Algorithm (SPEA2) in Java. Our implementation, Multiobjective Analyzer for Genetic Marker Acquisition (MAGMA), approximates the set of optimal trade-off solutions for large problems in minutes. This set is very useful for the design of large studies, including those oriented towards disease identification, genetic mapping, population studies, and haplotype-block elucidation.

Conclusion: Evolutionary algorithms are particularly suited for optimization problems that involve multiple objectives and a complex search space on which exact methods such as exhaustive enumeration cannot be applied. They provide flexibility with respect to the problem formulation if a problem description evolves or changes. Results are produced as a trade-off front, allowing the user to make informed decisions when prioritizing factors. MAGMA is open source and available at http://snp-magma.sourceforge.net. Evolutionary algorithms are well suited for many other applications in genomics.

Background

Geneticists are commonly faced with the task of selecting a subset of genetic markers from a large database. Geneticists then type these markers on a large number of individuals in disease-linkage or genetic-mapping studies. In general, the fewest number of markers necessary to achieve a particular goal are desired. A typical goal is the identification of a disease gene. Usually, the more markers selected for typing, the higher the probability of project success. However, selecting more markers entails additional cost. Therefore, a need exists for decision-support systems that allow geneticists to choose a set of genetic markers for a particular project that appropriately balances the need for project success with project cost.

Here, we present an evolutionary algorithm that supports genetic marker selection. This algorithm has been implemented in software, employed for large-scale single nucleotide polymorphism (SNP) selection projects, and is publicly available. We refer to the problem as one of SNP

selection for semantic convenience and to emphasize the current popularity of SNPs [1–3]. However, our discussion, algorithm, and software are equally applicable to most other genetic markers.

Other algorithms and tools for SNP selection are available. In general, these algorithms have specific constraints, are proprietary, or are designed for special instances of SNP selection [4,5]. For example, they may require (1) a small SNP library, (2) a known or partially known set of alleles, haplotypes, or haplotype blocks, or (3) a fixed cost or probability of assay success for each SNP. None of the available algorithms are multiobjective. Our algorithm, Multiobjective Analyzer for Genetic Marker Acquisition (MAGMA), provides simultaneous investigation of multiple balancings of different selection criteria. Our algorithm does not require any of the above constraints, but can incorporate them in situations where they are known.

Genetic markers occur at discrete positions along a genome, which is a collection of one or more linear chromosomes. Our discussion focuses on selection of markers across a single continuous linear span of the genome, or "locus". In practice, a locus is typically fifty kilobases to several megabases. Simple extensions permit our algorithm to encompass collections of disjoint linear or circular segments.

The probability of detecting an association between a marker and a disease phenotype decreases with distance between the marker and the actual position of the linked gene responsible for the phenotype. This probability can be affected by haplotype blocks, which have been defined as segments of a locus where this decrease is small or nonexistent [6,7]. The probability of detecting association is also influenced by other factors, such as the mutation rate of the marker.

If all markers are equivalent in value and cost, and if no prior knowledge is known about linkage relationships, then the optimal selection is to pick a subset of evenly spaced SNPs from the library, if such a subset exists. In this case, one can maximize the probability of detecting disease linkage by choosing markers as closely spaced as possible, using the project budget as a constraint on the number of SNPs selected. If information, or even partial information, on linkage relationships, such as the location or size of haplotype blocks, is known, then one economize by aiming to choose no more than one marker from each haplotype block. Even with perfect knowledge of haplotype blocks, some project designs will include more than one SNP per haplotype block to provide experimental robustness as well as protect against prior errors in haplotype-block determination.

In practice, all SNPs are not equivalent. SNPs in the database, or "library", will have been previously characterized to a lesser or greater extent. The library for a given project is likely to be drawn from public databases, such as dbSNP [8] and HGVbase [9], as well as in-house and commercial databases. In any of these databases, a SNP may be listed in error – in reality, there may be no SNP at that position in the genome. A SNP may be present in some genomes within a population of individuals, but so infrequently that it is of limited use for linkage studies. A SNP may be biochemically difficult to assay. All of these factors, and possibly others, which are either known or can be estimated, need to be considered during the selection of a set of SNPs from the library. We consider these factors weighted together as the "quality" of a SNP. Also, SNPs may have different costs. They may be an alternative type of genetic marker, be multiallelic, or require a different typing technology. They may be intellectual property, and have a licensing fee. We consider these factors weighted together as the "cost" of a SNP.

Our algorithm seeks to balance two optimization criteria: 1) the total project cost, and 2) the probability of detecting disease linkage. Modifications to the coding of our algorithm permit these criteria to be tailored to the goals of specific projects, including the possibility adding objective functions, substituting different objective functions, or splitting our objective functions into components. Our present discussion focuses on two objective functions, which produce solutions that are more easily visualized than solutions displayed in three or more dimensions.

The first objective, project cost, is the sum of the costs of each selected SNP. The second objective, the probability of detecting disease linkage, requires an arbitrary function that computes this probability from a set of SNP locations and qualities. This function can incorporate prior knowledge about gene locations and linkage disequilibrium, but such knowledge is not necessary. Use of this function requires little computation time and generates solutions that are useful in a variety of situations.

Evolutionary algorithms are suitable for exploring solution spaces of problems that are otherwise intractable; such problems have too many solutions to exhaustively enumerate, may be combinatorial in nature, and may be NP-hard. Alternative approaches include general problem-solving strategies such as dynamic programming or branch-and-bound as well as search algorithms such as linear programming and gradient descent. The advantage of these techniques is that they often guarantee generation of optimal solutions. However, as the complexity of models increase, it becomes more and more difficult, or even infeasible, to use them. Other general search techniques to tackle hard optimization problems, such as tabu search

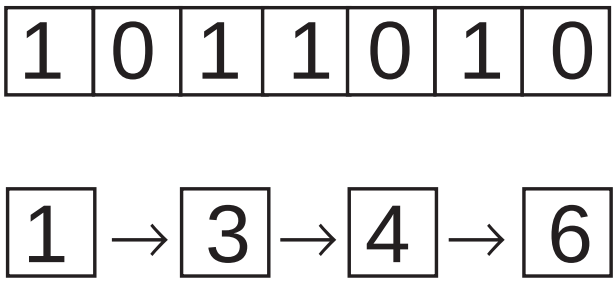

Figure 1
Genome Encoding. Alternative methods of encoding problem solutions as "genomes" include bit vectors (top) and arrays (bottom). In this cartoon, the solutions encoded by the two different methods are identical. For the problems we have studied, arrays are more compact and evolve more efficiently than bit vectors.

and simulated annealing, are designed for scenarios with a single optimization function. In contrast, evolutionary algorithms are well suited to handle both (1) a large search space, and (2) multiple objectives.

Evolutionary algorithms are excellent for exploring the set of all possible solutions, or solution space, of many multiobjective problems. In order to apply a multiobjective algorithm, solutions to a problem must be encodable as a genome (Figure 1). There may be more than one way to encode a solution; the choice of encodings can impact the performance of the algorithm. We illustrate this in our Results with an exploration of two possible encodings for our SNP selection problem. The first encoding represents solutions as a bit vector. The second encoding represents solutions as a variable-length list.

An evolutionary algorithm is initialized with a seed population of many random, or possibly non-random, solutions. The algorithm then steps through a number of iterations (Figure 2). During each iteration, the best solutions from the previous iteration are allowed to mutate and recombine. These new solutions then compete with each other, and with the previous best solutions. Good solutions are retained; poor solutions are rejected. Evolutionary algorithms work well when the operators for mutation and recombination are likely to efficiently explore the solution space. If good solutions to a problem are not reachable by mutation or recombination of other solutions, then an evolutionary algorithm will not be able to find these solutions. However, it is unlikely that any algorithm short of exhaustive search would be able to find such solutions.

Solutions to multiobjective problems are seldom unique. For example, if the two objective functions relate respectively to cost and performance, then the set of solutions will contain solutions with increasing performance and increasing cost. No solution, however, will have *less* performance for *more* cost. This set of alternative, optimal solutions, also known as the Pareto-optimal set, is thus a trade-off front (Figure 3). Typically, the user of the algorithm will examine the trade-off front produced by the algorithm and subjectively choose a single solution. This solution could be the best solution below a fixed cost, but often is chosen at a point of diminishing returns. The ability to identify points of diminishing returns is a major advantage to multiobjective optimization. Combining multiple objectives into a single objective in order to simplify a problem forces a decision to be made about the relative values of the different objectives before the trade-off front may be known.

The Pareto-optimal set contains those solutions that are not dominated by any other solution; we say a solution dominates another solution if the former is not worse in any objective and better in at least one objective. Accordingly, members of the Pareto-optimal set are referred to as non-dominated solutions or Pareto-optimal solutions. If we consider the image of the Pareto-optimal set in the space of objective function values, i.e., the set of the corresponding vectors of objective values, then we use the term Pareto-optimal front or trade-off front.

Evolutionary algorithms use a concept of "fitness" to decide which individual solutions will survive for the next generation. For single-objective evolutionary algorithms, fitness is typically identical to the single objective function. The innovation that evolutionary algorithms provide to multiobjective problems is their ability to define a real-valued fitness for each individual that is not only a function of how well that solutions fulfils each of the objective functions, but is also a function of all the other individuals presently in the population. The dependence on other individuals in the population is twofold: (1) individuals are more fit if they are closer to the front of best trade-offs within the population, and (2) individuals are more fit if they are in a sparser region of the optimization space. This innovation of population-dependency makes evolutionary algorithms fundamentally different from algorithms that operate merely by combining single objective functions.

Another major advantage of evolutionary algorithms is the ease of problem reformulation. In a rapidly changing field, such as genomics, the future use of algorithms may change unpredictably. Furthermore, subtle variations in the specifics of one problem instantiation may make inflexible algorithms less generally useful. In the example

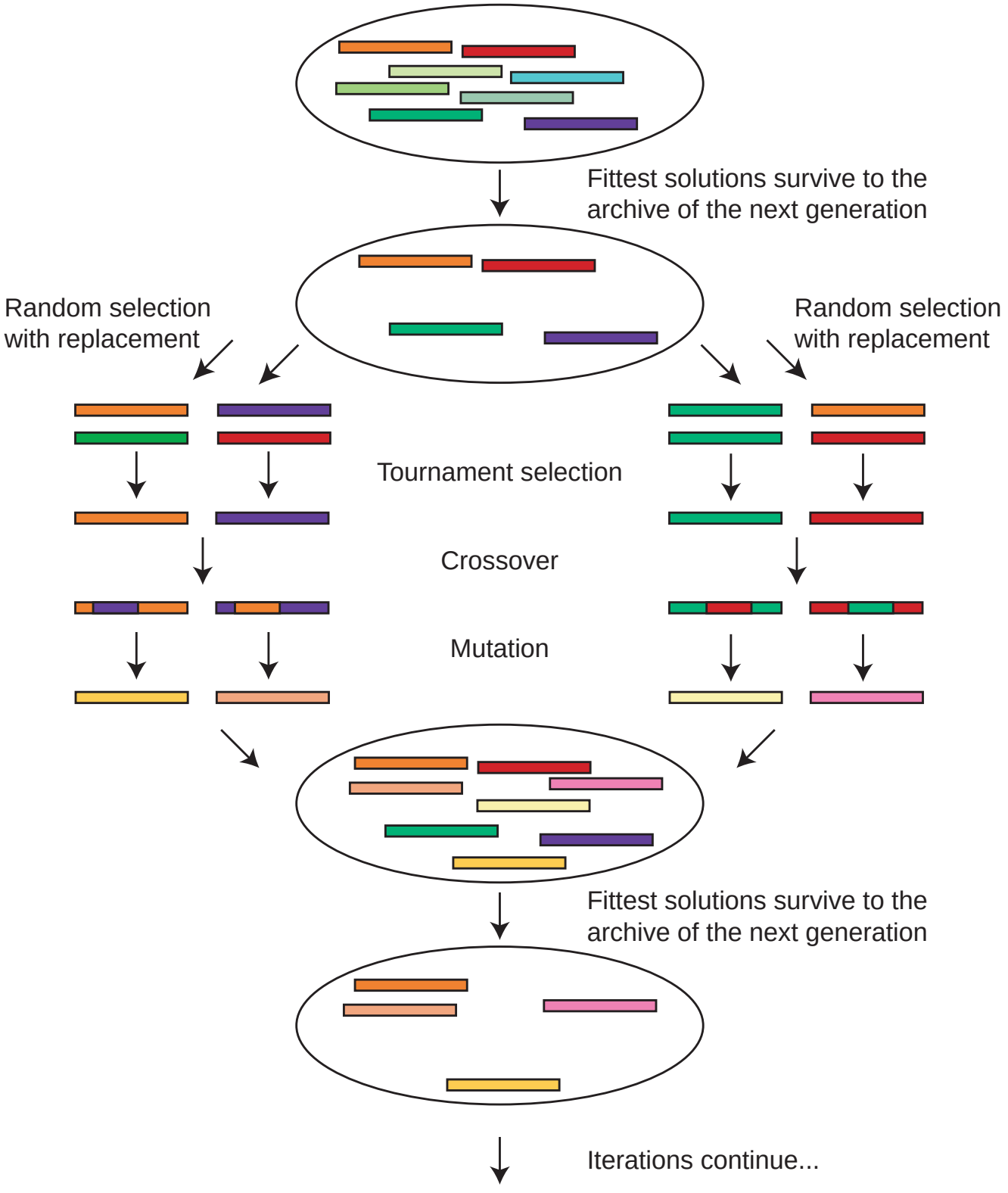

Figure 2
SPEA2. A cartoon of the SPEA2 algorithm, operating on a population size of eight with an archive of four. The algorithm continues for a fixed number of generations, and then outputs non-dominated solutions. Details of each of the steps are described in the text.

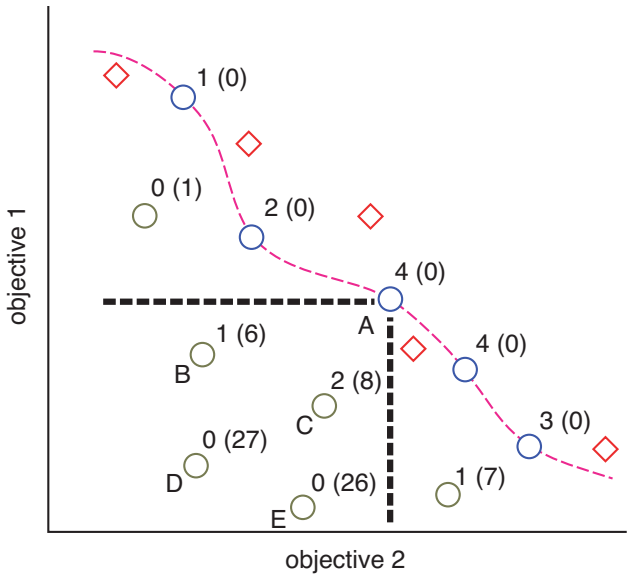

Figure 3
Pareto-Optimal Front. The two axes of the objectives form the objective space, which is two dimensional for the problems described in this paper. The positions of solutions on the Pareto-optimal front are shown in blue, connected by dashed pink. Each Pareto-optimal solution dominates all solutions to its lower left; for example, point A dominates points B-E, as bounded by the dashed black lines. Non-dominated solutions are shown as turquoise circles; dominated solutions are shown as olive circles. Strength is shown adjacent to each solution. Raw fitness is shown in parentheses. Because no raw fitnesses are identical, other than those on the front, density would have no impact on survival for this population as long as the archive size was at least five. Diamonds indicate positions in objective space of solutions not yet discovered by the algorithm, but that would be non-dominated once discovered.

we focus on in this paper, the objective functions of "cost" and "probability of disease detection" can both be represented in multiple different manners. For some of the more simple representations, dynamic programming would likely be competitive with our evolutionary algorithm. However, as these objective functions become more complex, perhaps including multiple interdependencies of the effects of linked SNPs, other algorithms may become untenable or require complete recoding of the software. Modification of the evolutionary algorithm typically entails only altering the objective function subroutine. Thus evolutionary algorithms are excellent choices for problems that are in development or are themselves evolving.

We provide an example of an evolutionary algorithm below, with commentary on our approach to its design and parameterization. We anticipate that this illustration of an evolutionary algorithm will prove useful as a reference for the design of evolutionary algorithms for other applications in genomics. Additionally, we describe the specifics of our software implementation, MAGMA. This software is open source and can be used directly for SNP selection; readers who are primarily interested in SNP selection may choose to skim our discussions of the details of our algorithm.

Our Results and Discussion include several examples from simulated data, as well as well as the application of our software to SNP selection in the human MHC locus.

Problem Formulation

A Simple Model

Ideally, one would like to optimize an objective function that directly represents the ultimate goal of a project, such as the probability of locating a disease gene. However, such objective functions may be subjective, difficult to describe, or require excessive computation. Proxy objectives may circumvent these difficulties. One simple and reasonable proxy is to search for evenly spaced high-quality SNPs, with an average spacing of s. This proxy objective would, in fact, be the truly desired objective if the target locus were describable in terms of haplotype blocks of constant length s, and if the locations of the haplotype block boundaries were unknown. Initially, we assume that all qualities are equal. An early version of this simple model is discussed in Hubley et al. [10].

We formally state the resulting optimization problem. Let n be the length of the locus under consideration and m be the number SNPs in the available library where each SNP i is described by two attributes:

p_i = position ($p_i \in \mathbf{N}$, $1 \le p_i \le n$)

q_i = quality ($q_i \in \mathbf{R}$, $q_i > 0$)

The attribute p_i denotes the position of a SNP. We here assume that the SNPs are ordered and unique according to the position (i.e., $p_1 < p_2 < \ldots < p_m$). The assumption that no two markers occur at the same position will be dropped for our more complex models. The quality of a SNP is represented by a positive real number q_i; a larger value stands for higher quality. Although it is not necessary to restrict q_i to the interval [0,1], it is often useful to do so, and interpret q_i as "the probability of a useful and successful assay for SNP i".

A solution to the problem, a non-empty subset of the available SNPs, can be expressed in terms of m decision

variables $x_i \in \{0,1\}$ with $x_i = 1$ if and only if SNP i is in the solution. For convenience, we introduce the variables x_0 and x_{m+1} which are by definition set to 1 and refer to two fictive SNPs that mark the left (position 0) and the right end (position $n + 1$) of the locus. Now, consider the deviation d_{ij} of the spacing between two adjacent SNPs from the optimal spacing s:

$$d_{ij} = \left(s - | p_j - p_i | \right) \cdot x_i \cdot x_j \cdot \prod_{k=i+1}^{j=1} \left(1 - x_k \right)$$

If the SNPs i and j are immediate neighbors in the selected SNP subset, then d_{ij} gives the number of base pairs between them; otherwise, d_{ij} equals zero. We chose our proxy goal for this simple model to minimize the mean squared deviation from the ideal gap length s

$$f_1 \left(x_1, x_2, ..., x_m \right) = \frac{1}{1 + \sum_{i=1}^{m} x_i} \sum_{i=0}^{m} \sum_{j=i+1}^{m+1} d_{ij}$$

while also maximizing the average quality

$$f_2 \left(x_1, x_2, ..., x_m \right) = \frac{1}{\sum_{i=1}^{m} x_i} \sum_{i=0}^{m} q_i \cdot x_i$$

In some cases, such as to eliminate the possibility of large gaps in solutions, it is useful to add additional constraints. For this simple formulation, we add the constraint that all gaps are less than or equal to s:

$\forall\ 0 \le i \le m\ \forall\ i < j \le m + 1$: $d_{ij} \le s$

For certain problems, it may not be possible to fulfill the constraint. There might be a pair of SNPs i and j with $d_{ij} > s$ even if all m SNPs are selected. In this case, the problem can be divided into subproblems that can be solved independently. Alternatively, the algorithm can allow solutions to have constraint violations as long as there are not an excess of competing solutions without violations.

Our implementations permit problems to be subdivided if large gaps are present that divide the locus into two or more independent optimization problems. Solutions are thus independent, with multiple trade-off fronts produced. If a single trade-off front for the entire locus is desired, then it may be more convenient not to subdivide a locus with large gaps. This single trade-off front could also be generated by combinatorially combining the trade-off fronts form each independent solution. Such combinatorial combinations may be superior at approximating the Pareto-optimal set, which is the set of solutions that cannot be improved in any objective without degradation in another.

Our initial implementation encoded solutions as bit-vectors of length m. Bit i represented the presence or absence of a SNP at position i of the locus. Allowed variations were bit flip mutations and recombination. A typical bit-flip frequency was 4% per bit position. A typical recombination frequency was 80%. Therefore, on average, each solution in the population is modified at one position. Because the solutions we sought tended to be sparse (most of the bits set to zero), our choice of encodings for solution genomes created considerable overhead, both in storage and in time exploring regions of the solution space of little interest. In particular, because a bit flip was equally likely at all positions regardless of their current state, solutions were biased towards an equal number of selected and unselected SNPs. This bias could have been altered towards a more interesting equilibrium by separating the $0 \rightarrow 1$ mutation rate from the $1 \rightarrow 0$ mutation rate. However, a superior solution is to change the encoding of the genome, which we describe in the next section. We draw attention to the choice of genome encodings because such choices can considerably impact the performance and success of an evolutionary algorithm.

The simple model is unsatisfying in a number of respects. Notably, average quality does not discriminate between different distributions of quality SNPs – solutions with high quality SNPs evenly dispersed are different than solutions where all the high-quality SNPs are clustered. Also, solutions with a number very high quality SNPs and an equivalent number of low quality SNPs may be different than solutions with the same total number of SNPs of intermediate quality. The two objective functions, corresponding to "deviation from ideal gap length" and "average quality" are somewhat non-intuitive. The average quality is effectively acting as a proxy for total cost – the Pareto-optimal solutions with high average qualities will tend to have few SNPs, and the solutions with low average qualities will tend to have many SNPs. For this reasons, the trade-off front produced is somewhat difficult to interpret. However, in practice, although somewhat unsatisfying, the simple model does produce reasonable solutions to real problems.

A Probabilistic Model

The simple model above is useful, but can be improved in a number of manners. In particular, it sidesteps the true objectives of a project with the proxy objective functions of "spacing deviation" and "average quality". Objectives that more realistically reflect an actual project's objectives would reflect "cost" and "probability of project success". The first objective function "cost" is straightforward to model. We prefer to conceptualize the problem as a maximization, so we use a cost function inversely proportional to the number of SNPs in a solution:

$$f_1(x_1, x_2, ..., x_m) = \frac{1}{1 + \sum_{i=1}^{m} x_i c_i}$$

where c_i is the cost associated with SNP i. A number of difficulties arise if the costs are not independent. For example, if experimental protocol dictates that SNPs be processed in 96-well trays, the cost for typing ninety-six SNPs may be the same as the cost of typing a single SNP, but half the cost of typing ninety-seven SNPs. This particular complication can be addressed by post-processing the trade-off front. For the remainder of this paper, we assume the cost per SNP is constant and equal to one.

The second objective function we propose here makes a number of compromises between computability, subjectivity, and genetic reality. However, the exact form of the function can be altered to meet other compromises. Our modular software implementation facilitates such alterations.

To motivate our model, we make a number of assumptions, while recognizing that the resulting model may be quite useful as an approximation even if these assumptions are not met. We assume that a sought-after disease gene exists in the locus in question, and that it can be linked to a single base position. We consider the probability of linking a trait to any given SNP to be a function only of the quality of that SNP and of the distance of the gene position from that SNP. We consider the gene will be "identified" if it is detected to be linked to at least one selected SNP. We assume the prior probability of the disease location is constant across all base pairs in the locus.

We treat the quality of a SNP as the probability the SNP successfully detects linkage at its own location, and that this probability decays linearly to a minimum of zero as a function of distance from the SNP location. From a modelling point of view, the quality is thus a heuristic combination of (1) allele frequency, which impacts statistical inference of linkage, (2) database reliability, which relates to the probability that a SNP may not actually exist, and (3) biochemical suitability, which determines the probability of successful assay implementation. As discussed above, these components could be separated into separate objective functions, but such separation would require departure from two dimensionality. The probability of SNP i being successfully linked to a disease trait at base pair position h is thus:

$$\pi_{ih} = x_i \cdot q_i \cdot \left| \frac{\max(0, s - |p_i - h|)}{s} \right|$$

The assumption of linear probability decay with distance is motivated largely by computational tractability. It is also a reasonable assumption if little or no prior knowledge about linkage relationships across the target locus exist for the study population. The "probability of project success" objective function is thus:

$$f_2(x_1, x_2, ..., x_m) = \sum_{h=1}^{n} \frac{\left(1 - \prod_{i=1}^{m} (1 - \pi_{ih})\right)}{n} \qquad (1)$$

In practice, for computational speed, we approximate f_2 by sampling values in the sum once every few hundred base pairs. Because typical values of s are at least an order of magnitude larger than the sampling distance, such an approximation is more than adequate for purposes of the evolutionary algorithm.

There are a number of possible extensions to f_2, which come with varying degrees of increased coding complexity or computational overhead. A number of these extensions arise by removing assumptions discussed above. For example, the prior probability of the disease location may not be constant, and may, for example, be influenced by previous calculations of linkage-disequilibrium scores across the region. These prior probabilities can be incorporated into the objective function, allowing for improvements in the selected SNP set. Knowledge of specific linkage relationships, particularly of known haplotype-block locations, can also be included in the model.

In the algorithm implementation, we encode the solution as an array of variable length. Each position a in each array contains a location p_a, with $p_1 < p_2 < \ldots < p_A$. This encoding of the genome as an array uses less memory than a bit vector and produces mutations that more likely to improve solutions. Allowed variations are insertions, deletions, substitutions, and recombination. A typical mutation frequency is 13% per array position per solution. If a mutation occurs, the mutation operator is chosen with the following frequencies: insertion – 40%, deletion – 10%, or substitution – 50%. An insertion adds a position to the solution array immediately before or after the mutated array position a. The SNP position placed in the new array position is picked uniformly from all unpicked SNPs between p_{a-1} and p_{a+1}. If there is no such SNP, then the mutation has no effect. A deletion deletes array position a if the resulting deletion does not create a constraint violation; otherwise it is ignored. A substitution replaces the SNP position at array position a, with a SNP uniformly drawn from the unpicked SNPs between p_{a-1} and p_{a+1}; substitutions creating constraint violations are ignored. A typical recombination frequency is 70%. Recombination takes place by choosing two SNP pair positions a and b uniformly from a solution and exchanging the array positions with SNPs from another solution where $a' >= a$ and

$b' <= b$. Multipoint recombination is also allowed, as a user option.

Results

Simple Model

The simple model was tested on a 90 kb segment of the human major-histocompatibility locus. The library contained 626 SNPs. The trade-off front generated by the evolutionary algorithm after 200 generations is depicted in Figure 4. The density of solutions increases as the first objective increases. This illustrates the structure of the solution space for this particular problem. The heuristic solution represents a trade-off that neither dominates nor is dominated by any MAGMA solution, and is located in the middle of the front rather than on one of its extremes. The best solution in the first objective contains only 35 SNPs with an average quality of 117. The other extreme solution includes 85 SNPs and achieves an average quality of 181; basically all high quality SNPs are chosen and the large gaps are filled by SNPs of lower quality.

Although these solutions were deemed acceptable, we did not do extensive further testing or comparisons of the simple model due to the drawbacks of the model and its implementation, discussed in Problem Formulation.

Probabilistic Model

We tested MAGMA against several benchmarks, and employed it for several real SNP selection problems. First, we discuss performance against benchmarks. For the first benchmark, we selected a region from the human MHC locus, selected so that the SNP library was small enough that all possible solutions could be evaluated with an exhaustive enumeration. Therefore, the exact Pareto-optimal set was known.

The first twenty SNPs from a much larger library of SNPs derived from the MHC were chosen as a representative library. We attempted exhaustive enumeration on larger libraries but were foiled by the excessive time required for computation. For this benchmark, serial enumeration required seven hours and ten minutes; MAGMA finished its computations in one minute and twenty-eight seconds (Figure 5).

The disadvantage of this benchmark is that it illustrates the performance of MAGMA on easy problems, which are also adequately solved by other algorithms. A more appropriate benchmark is one for much larger SNP libraries. Unfortunately, exhaustive solutions from larger libraries are not always tractable. Therefore, we carefully constructed a large library with a vast number of SNPs but with an analytically demonstrable optimal region of the Pareto-optimal front. MAGMA has no particular reason to seek out this portion of the front any faster than if it were not intentionally designed into the library. Therefore, we judge this constructed library to have been capable of providing an excellent benchmark for multiobjective optimization.

In order to create a benchmark for MAGMA with an input dataset similar in magnitude to those that we anticipate for actual biological problems, we intentionally created a dataset with 334 SNPs of quality 1.0 spaced evenly across a 1 Mb locus, with one thousand additional SNPs of quality 0.01–0.05 with uniformly distributed positions across the same locus. Although we cannot determine the entire Pareto-optimal set for this library, we know that one point on the front must consist of the set of high-quality SNPs. Without heuristic seeding, MAGMA's algorithm should not find this analytically discoverable solution any faster than any other optimal solution. Thus, we judged this to be a good benchmark. MAGMA discovered a front with this solution in five hours and forty-one minutes, after thirteen thousand generations. A solution with 344 SNPs was present after only one thousand generations; a solution with 337 SNPs was present after 3500 generations. To test the robustness of the algorithm to the random seed, we fixed the number of generations at twenty thousand and ran ten independent additional optimizations. The seeded solution of 334 SNPs was recovered after each optimization. The Pareto-optimal fronts outputted for each of these optimizations were nearly indistinguishable (data not shown).

The region of the human MHC locus illustrated in Figure 6 provides an example of a real SNP-selection problem. The trade-off front produced by MAGMA is graphed in Figure 7. The set of SNPs chosen for further study was selected by visual evaluation of this graph. For the development of biochemical assays, a solution from a region of diminishing returns was selected that had a cost compatible with available resources. The solution produced by the heuristic lies just behind the front produced by MAGMA. The heuristic is prevented from identifying a solution on the Pareto-optimal front largely because of its reliance on binning quality values (see Methods). Nevertheless, the solution produced by the heuristic is quite good.

Discussion

MAGMA is the first publicly available software for SNP selection in the absence of haplotype information. MAGMA is the first such software with a published algorithm, and the first with open-source code. The majority of SNP-selection tasks are done with no prior haplotype information, so MAGMA should be useful for the majority of SNP-selection tasks. As the first in its class, MAGMA should provide a useful benchmark for future SNP selection software. Software is available for specialized cases of

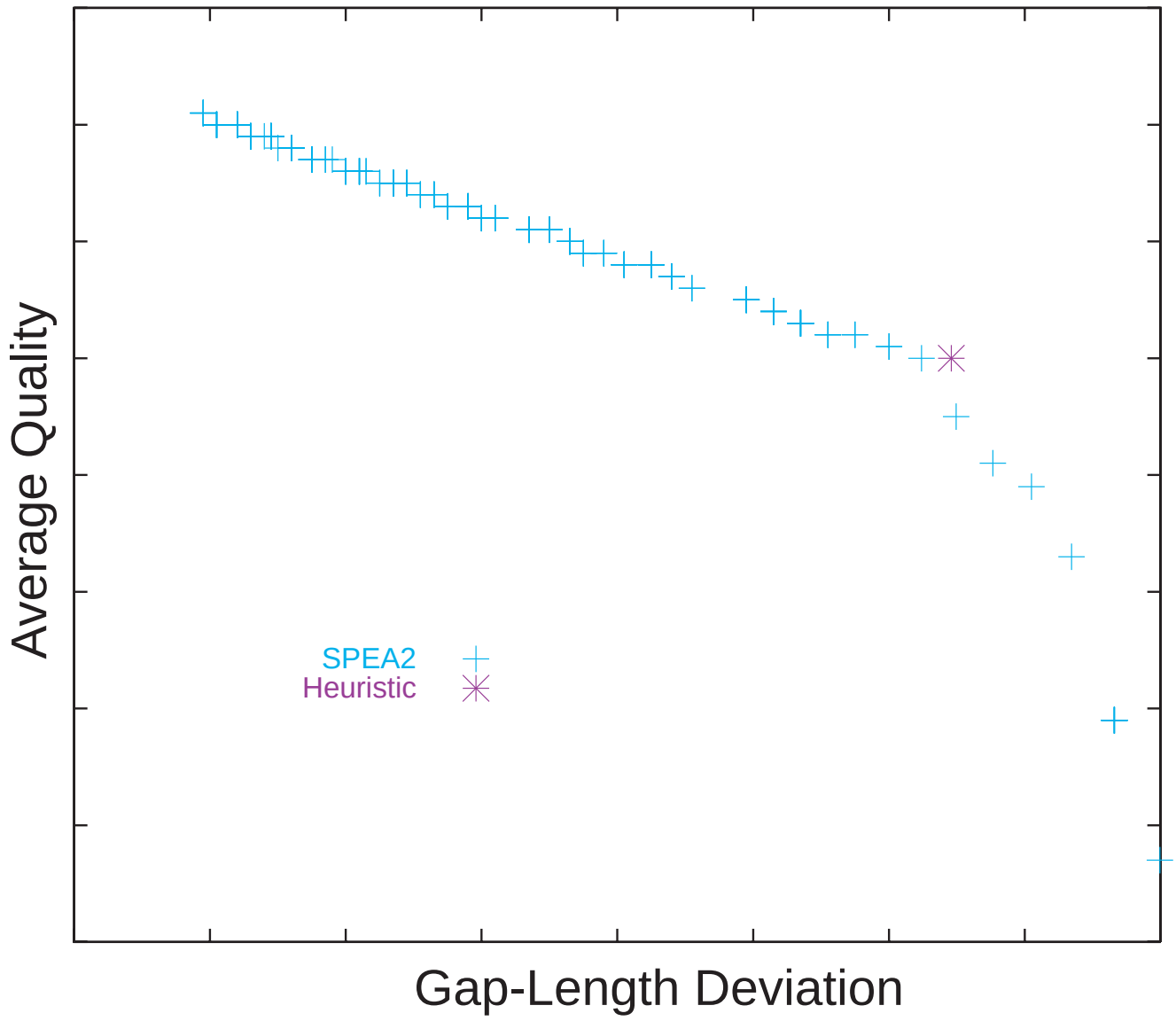

Figure 4
MAGMA Output. The trade-off front obtained from MAGMA after six hundred generations for the simple problem formulation. In this case, the heuristic produces a solution that is neither dominated by nor dominates any of the solutions offered by MAGMA. Slight unevenness in the front is likely due both to the structure of the SNP library, resulting in limited choices for solutions, and to suboptimality in the trade-off front. The abscissa is reversed to place the Pareto-optimal front to the upper right for visual consistency with the other graphs in this paper (f1 in the simple problem formulation is minimized). The scales of the axes are arbitrary.

SNP selection, such as when "haplotype-tagging" SNPs are desired [[11,12], and software by David Clayton at http://www-gene.cimr.cam.ac.uk/clayton]. We believe that, in the absence of very complete data on the population frequencies and haplotypes of all SNPs in a region, flaws could be introduced into a study design that relies on the concept of haplotype blocks. Therefore we believe that MAGMA will often be more useful than such specialty software. Also, the algorithms in specialty software are often limited to SNP libraries containing no more than several dozen SNPs.

MAGMA also represents one of the first applications of a genetic algorithm to a problem in genomics. Our

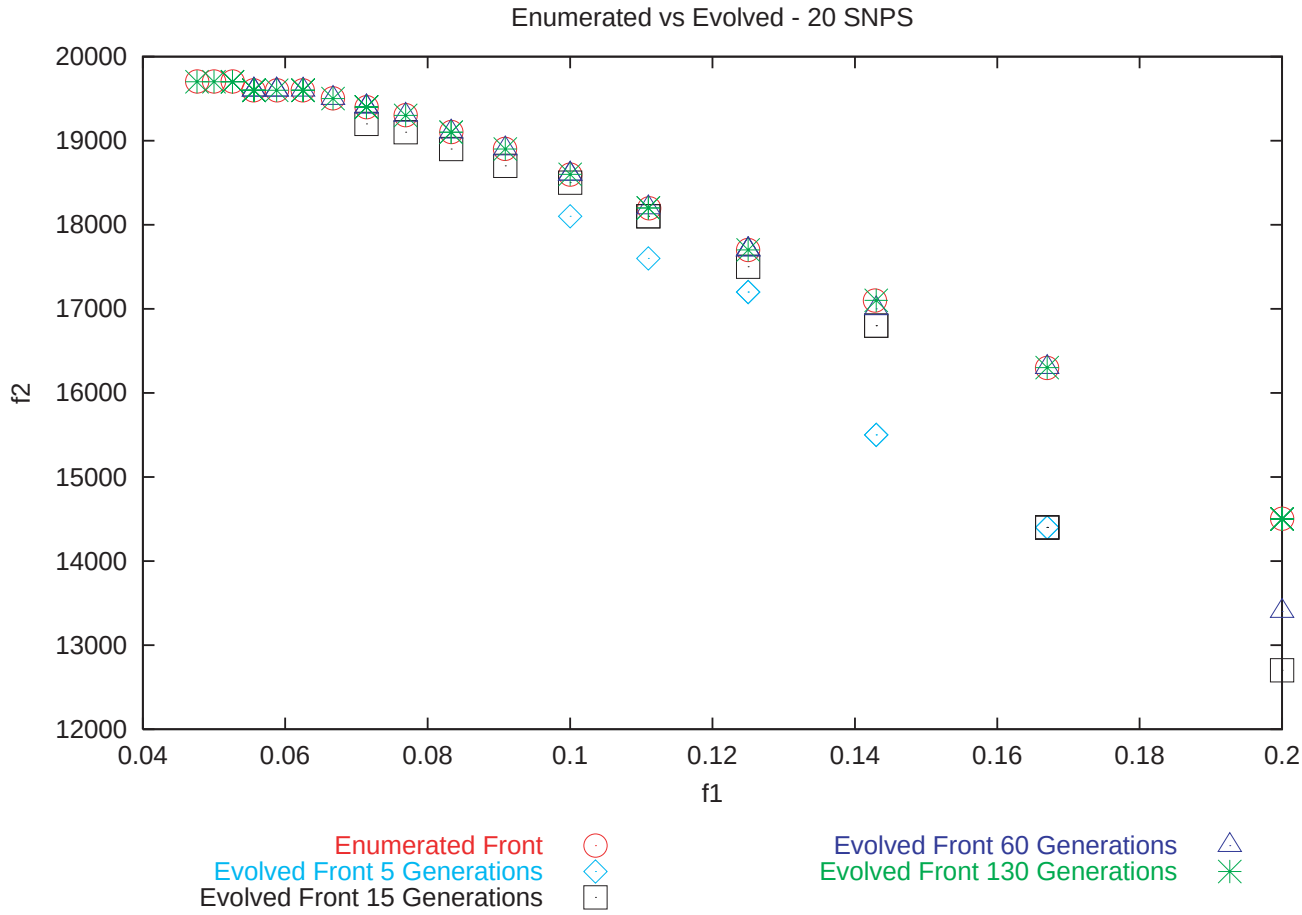

Figure 5
Enumeration Benchmark. We exhaustively enumerated all solutions for SNP selection from a library of twenty SNPs. We then tested MAGMA's ability to identify the optimal front on this same set of SNPs. Prior to 130 generations, MAGMA discovered the entire Pareto-optimal front. Enumeration required seven hours on a desktop machine; MAGMA computed the 130 generations in ninety seconds. The two objective functions, f1 (inversely proportional to the number of SNPs in the solution) and f2 ("coverage"), are expressed in arbitrary units.

applications of MAGMA demonstrate the particular usefulness of evolutionary algorithms in the presence of multiple optimization criteria. Firstly, an evolutionary-based approach allows generation of a set of trade-off solutions which provide additional information about the problem, such as the magnitude of the conflict between objectives, whether there are many or few potential solutions, and the structure of the search space. Knowing which alternatives are available can strengthen the confidence in the choice of a particular solution. Secondly, an evolutionary algorithm provides flexibility. Additional objectives and constraints can be incorporated with only little programming effort. For instance, in the future we may split our single quality objective into several.

We speculate that evolutionary algorithms may be useful for other genomics applications, such as haplotype-block partitioning [13]. They also have been used to aid structure prediction [14]. This utility makes intuitive sense, as many problems in genomics have solutions that are easy to encode as a virtual genome – particularly because many of these solutions are actual representations of a real genome. Likewise, the mutation operators that an evolutionary algorithm employs on its solutions often parallel the mutations that occurred naturally during the biological evolution of the system being modelled. Our speculation that genetic algorithms may have utility in other genomics applications may not apply to all genomics problems.

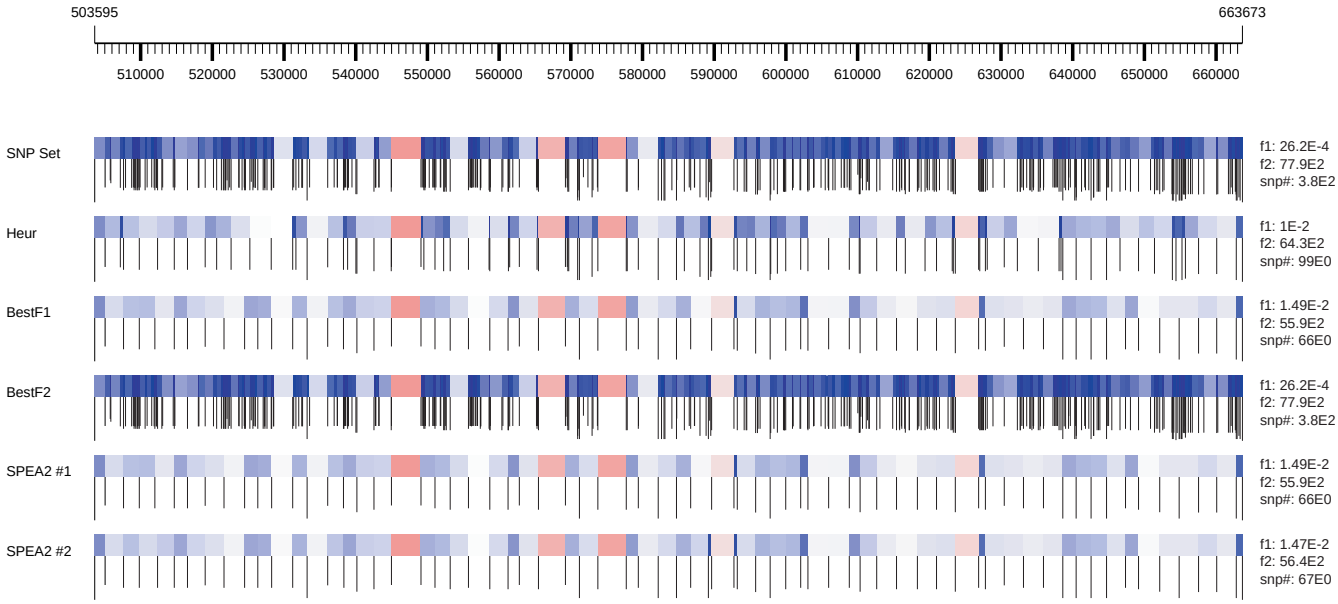

Figure 6

MHC locus. Illustration of MAGMA on a non-contrived problem. The scale at the top is number in base pairs relative to an arbitrarily designated beginning of part of the human MHC region. This locus is flanked by large gaps without known SNPs, so is a natural span for input to MAGMA. The library consists of 382 SNPs of varying quality. SNPs are indicated by black vertical lines; quality is proportional to the length of the lines. The colors of the bars indicate departure from a user-defined SNP influence radius, in this case, 6000 bp; red areas have sparser coverage while blue areas are more densely covered. The number of solutions displayed (in this case, the best f1 and f2, and two others) can be set by the user. "Heur" indicates the solution seeded by the heuristic. Numbers to the right indicate the value of each objective function for the solution, as well as the number of SNPs in the solution.

Exact optimization algorithms can, with difficulty, be tailored for specific sets of objective functions. Generally, this requires that the objective functions be relatively simple. For example, it is difficult to design a dynamic algorithm that can efficiently handle linkage dependencies between non-adjacent SNPs (e.g. SNPs linearly arranged A-B-C, with A linked to C but not to B). However, many practical and reasonable implementations of SNP selection will not need or have available such complex linkage information. We anticipate that exact optimization algorithms could be designed for these cases. For the simple models that we have described above, we anticipate that dynamic algorithms that run in pseudo-polynomial time could be constructed. Even these algorithms, however, could be slow for certain problems, such as those with exponentially many solutions in the Pareto-optimal set.

Deterministic heuristics tailored to the application at hand often produce reasonably good results if sufficient problem knowledge is available. However, the design of these heuristics gets more difficult as more objectives are involved, and the single solution produced does not provide information about alternative solutions. One way to tackle complex multiobjective optimization problems is to combine both approaches into a single algorithm, by using the results of heuristics as seeds for an evolutionary algorithm.

Genomes are not always well described by linear models. For example, in the human genome, there may be hundred-kilobase regions present in some individuals but not in others. There are at least three solutions to this difficulty. The first solution is to represent the genome as a series of independent regions and treat these regions independently for SNP selection. The second solution is to concatenate these independent regions into a representative genome that includes all possible segments that might be found in any individual. The concatenation can be done in a manner that tends to preserve adjacencies found in the majority of genomes in the study population. The third solution is to recode an optimization function in a manner that does not assume a linear model. The possibilities for recoding are vast, but might incorporate, for example, a knowledge of population frequencies of certain rare large-scale polymorphisms. An evolutionary

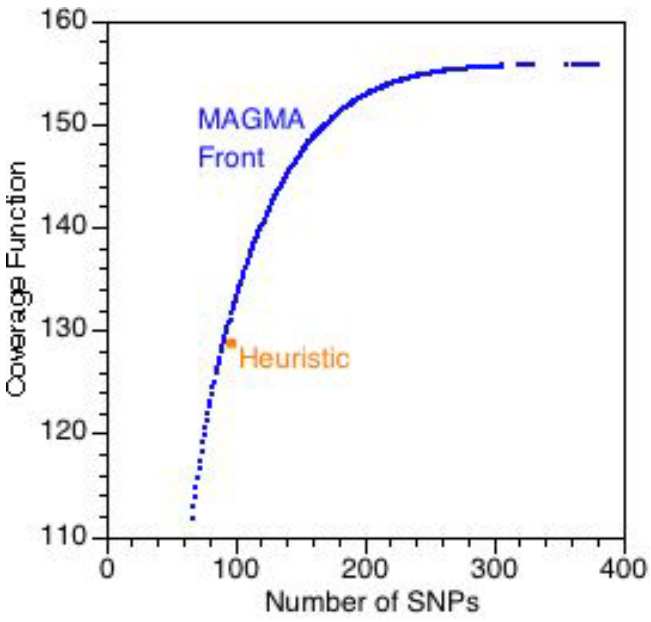

Figure 7
MHC Trade-off Front. Illustration of MAGMA on a non-contrived problem, a portion of the human MHC region. The approximation to the Pareto-optimal front produced by MAGMA is likely to be exact (see the description of the benchmark tests in the text). The solution produced by the heuristic (position in optimization space displayed by a yellow square) was used as a seed. The coverage function is expressed in units proportional to the probability of linkage detection (equation 1). The production of a trade-off front, such as this one, is a major advantage of using a multiobjective algorithm.

algorithm should work well even with such a complex optimization function.

As the input becomes very large, MAGMA takes considerable time finding the best solutions, and may even bog down. There are several approaches that can push this problem-size boundary back. The first is to break the problem up into subproblems. The second is to seed the initial population with the results of heuristics – possibly including concatenations of MAGMA outputs from subsections. The third is to alter the mutation parameters. We have discovered that increasing the crossover frequency tends to speed convergence for large problems. However, even with such struggles, problems can be made too large even for MAGMA. This problem size for an average personal computer is currently about ten thousand SNPs in the library. In our experience, if particular types of solutions are sought by the end-user, such as sparse solutions, altering parameters, such as increasing the deletion probability, can speed the algorithm in finding good solutions in this region of the solution space by an order of magnitude or more. However, MAGMA runs so fast without this speedup that we feel that it is seldom worth effort to seek out optimal parameterizations; the robustness of the genetic algorithm allows it to explore vast solution spaces even with widely different parameterizations.

There may be other approaches that would help tame larger problems, if such problems arrive on the genomics scene. We have considered, but have not explored, the possibility of adding "cataclysmic" events, possibly operating in the evolutionary algorithm in a manner similar to what an asteroid might have done to the dinosaurs. By destroying much of an archive, or by adding a new infusion of very different genetic material, such events might jar the algorithm out of local optima. The frequency of these events could be set to diminish as the algorithm proceeded. More simply, mutation rates could be set very high initially, and trailed off as the algorithm proceeded. Also, very large problems can be subdivided, with each subproblem treated independently. Even if no effort were made to optimize the regions where the edges of solutions met, if each solution consisted of thousands of selected SNPs, then it would seem unlikely that any cost inefficiency due to edge effects would exceed a fraction of one percent.

The choice of quality and cost are decisions that must be made by users of MAGMA. We have not explored non-uniform costs, as all the SNPs we use have approximately the same cost to implement. Even though we have not explored non-uniform costs, we believe it would be straightforward accounting to add these in, where such costs are known or can be approximated. The choice of quality in practice is likely to be subjective, unless the user has access to unusually complete data relating to the probability of assay success. We believe, based on the problems we have examined, that MAGMA's solutions are robust with respect to moderate differences in assigning quality to SNPs. For SNPs from public databases, such as dbSNP, our default qualities are derived from the validation annotation associated with a SNP entry, such that validated SNPs with frequency information receive a quality of 0.95 and SNPs with no validation receive a quality of 0.5. We alter these initially assigned qualities slightly based on properties of the flanking sequence as well as the presence of nearby polymorphisms. These qualities are currently assigned by situational PERL scripts.

Much current research focuses on protein-coding regions. Researchers may be less interested in a uniform distribution of SNPs across a locus, and more interested in a distribution with SNPs concentrated in protein-coding

regions. The most natural solution is to define a prior probability across the locus for the location the disease-causing mutation. For example, one can assume that a disease is ten times more likely to fall on a particular protein-coding base than on a particular non-coding base. These prior probabilities can then be incorporated into equation (1), requiring a small change to MAGMA's current hard code. There are also three heuristic approaches to biasing SNP selection to select regions. The first, which we use most commonly in practice, is to boost the quality scores of SNPs in protein coding-regions. This heuristic, in practice, often achieves the desired effect. For example, doubling the quality scores of SNPs in protein-coding regions produces concentration of SNPs in these regions in the MAGMA output. The second approach for biasing SNP selection is to separate the protein-coding regions from the non-coding regions prior to running MAGMA, and then to evaluate the output from each of these regions separately. This gives the researcher maximum control over the choice of SNP density in each region, but at the cost of more user intervention. The third approach to bias SNP selection is not to use MAGMA at all for protein coding regions, but only for non-coding regions. If, which is rarely the case, all alleles of the protein are known, then it no longer makes sense to choose a uniform distribution of SNPs. Rather, it makes sense to choose a minimal set of SNPs that discriminates between these alleles. For this purpose, we use a suite of in-house software tools, collectively dubbed "Haplotype Resolver". Haplotype Resolver is currently in development.

Our implementation, MAGMA, could have applications outside of genomics. It should be useful in many situations where a subset of linearly arranged objects is desired. These situations could occur, for example, in gardening – thinning a row of carrots, manufacturing – eliminating items from an over-concentrated assembly line, and air traffic control – choosing a subset of airborne planes to enter or leave a landing queue.

Graphical representation of solutions is particularly important for multiobjective optimization problems. Multiobjective optimization produces a population of solutions from which a user must subjectively choose a final solution. MAGMA offers several output formats, although others may be envisioned. Graphical representation is the major constraint limiting the number of different objective functions that can conveniently be used. Thus, if one has a practical three-dimensional graphical visualization interface, one could easily produce and display Pareto-optimal fronts of three objective functions. If the evolutionary algorithms are being used in a high-throughput system, the subjective decision made by the human may be automated without necessarily losing the advantages of the multiobjective solution. For example, the "solution choosing" algorithm might be trained to select the best point of diminishing returns along the trade-off front.

Conclusions

We have implemented an evolutionary algorithm, MAGMA, that produces useful solutions to the problem of selecting SNPs for use in genetic mapping studies. Evolutionary algorithms are likely to have general utility for a variety of problems in genomics. Due to the ease of recoding objective functions, they are particularly useful in the development phase of projects, when problem descriptions may change frequently.

Methods

Strength-Pareto Evolutionary Algorithm

The original Strength-Pareto Evolutionary Algorithm (SPEA) and an improvement, SPEA2 [15,16], were described and implemented by Zitzler et al. This algorithm was shown to provide excellent performance in comparison to existing methods [17]. Luke [18,19] provided an implementation of SPEA2 in Java. We redesigned an implementation in Java for MAGMA. Our designs are based in part on a number of overviews of evolutionary algorithms [17,20–25]. Effects of density, archiving, and elitism are discussed in Laumanns et al. [26].

An outline of the algorithm follows, where "individual" is used synonymously with "individual solution" (see also Figure 2):

1. Set $t = 0$.

2. Generate the initial population, P_0, and archive, A_0.

3. While $(t < T)$ {

   a. Calculate fitness of all individuals in P_t and A_t.

   b. Set A_{t+1} = non-dominated individuals in P_t and A_t.

   c. If $A_{t+1} > N$, then reduce A_{t+1}, else fill A_{t+1}.

   d. Fill mating pool by binary tournament selection with replacement on A_{t+1}.

   e. Apply variability operators to mating pool and place results in P_{t+1}.

   f. Set $t = t+1$.

}

4. Output the non-dominated set of A_{t+1}. A non-dominated solution is at least as good in all objectives, and

possibly better in at least one objective, than every other solution.

We typically set the population size, *N*, to five hundred, and the archive size to half the population size. *T* is a user-defined integer specifying the number of generations. Fitness is defined as raw fitness plus a density factor. In SPEA2, a lower fitness value is better than a higher fitness value. This is the opposite sign convention compared to usual biological nomenclature. Raw fitness is the sum of the strengths of all the dominators of an individual. The strength of an individual is equal to the number of individuals that individual dominates (see Figure 3). The density factor increases in regions of the optimization space that are dense [26]. The use of a density factor tends to push the edges of the population of solutions towards unexplored regions of the solution space. The density factor is small enough, with $d \in [0,1)$, so that it only serves to discriminate between individuals with identical raw fitness. Since all non-dominated individuals have a raw fitness of zero, they will never be eliminated due to overcrowding unless the size of the archive is set to be smaller than the size of the Pareto-optimal set. We therefore recommend that the archive size, which is a user-defined constant, be set large enough to contain the entire front, or the front with sufficient granularity to elucidate trade-offs and points of diminishing returns. Also, the density factor can be affected by arbitrary scaling of the two objective functions with respect to each other. If the absolute value of the differences of the two objective functions of nearby solutions is large, then density may serve primarily to eliminate solutions that are "dense" only in the dimension of the dominant objective function. This density bias can be avoided by scaling the objective functions appropriately, but in practice this is not necessary, as such scaling has little effect on the performance of the algorithm.

Density is inversely related to the Euclidean distance in the optimization space to the *k*th nearest neighbor, where *k* is a user defined constant. A common choice of *k* is the square root of the population size, but setting $k = 1$ greatly speeds computation with no significant effect on the performance of the algorithm. Therefore SPEA2 currently uses $k = 1$. Density is recomputed after each elimination.

Binary tournament selection draws two individuals from the archive, and takes the most-fit individual. This process is repeated to produce a second individual. These two individuals then undergo crossovers at zero or more sites, with a probability of at least one crossover around eighty percent. The two products of the crossovers then individually have a probability of mutating at every position. The resulting two individuals are placed in the population of the next generation. Tournament selection then continues until the entire next population is formed.

MAGMA incorporates several details in implementation that are absent in the original SPEA2 algorithm. Notably, as long as there are unique solutions, MAGMA fills its archive with unique solutions before placing any duplicates in the archive. This may decrease the frequency at which more fit regions of space are explored, and increase the frequency at which less fit regions are explored. However, it also may provide a mechanism to escape local optima. The issue of duplicate solutions tends to arise only in relatively simple problems, many of which can be solved by exhaustive enumeration.

The choice of which individuals to place in a seed population tends to affect the number of generations required for MAGMA to converge towards a near-optimal front, but not to affect the quality of the front. There is a possibility that placing individual solutions that are already nearly optimal in the front might bias a large portion of the population towards a particular local optimum. To guard against this possibility, the initial seed population should consist of random individuals. However, since the algorithm seems to converge more quickly without loss of optimality on the SNP selections we have encountered to date, we include in the original population a single individual from the output of a heuristic algorithm, and fill the remaining initial population randomly.

MAGMA includes the option to place constraints on solutions, in addition to judging solutions based on the two objective functions. For example, a researcher may not be interested in any solution that contains more than one hundred SNPs, but may be interested in exploring the trade-off front of all solutions with fewer SNPs. One can constrain MAGMA to reject solutions with more than one hundred SNPs. This is implemented by allowing individual solutions in the population to violate any constraints. However, individuals that do not violate any constraints will always dominate individuals with constraint violations. Typically after a few generations, all individuals in the archive will satisfy the constraints. Adding constraints may impede the algorithm's ability to escape local optima, or may slow convergence. We have explored the performance of the algorithm with a single constraint. For the SNP selection problems we have encountered, convergence is much faster because the algorithm does not spend time exploring regions of the solution space that are far from the regions of our interest. Another advantage is that the output of the algorithm is focused on solutions of interest. This advantage of a more focused output could also be obtained through post-processing, but at a cost of reducing the number of proffered solutions.

Our implementation, Multiobjective Analyzer for Genetic Marker Acquisition (MAGMA), is available at SourceForge http://snp-magma.sf.net.

Heuristic Algorithm

We designed a heuristic algorithm as a foil to MAGMA's evolutionary algorithm. This heuristic algorithm produces solutions that approach or equal Pareto-optimal solutions found by MAGMA. However, the heuristic produces only a single solution, not a trade-off front. The heuristic's strategy is to find a set of evenly spaced markers of as high a quality as possible. The algorithm first divides the qualities into three (or optionally more) classes, and subsequently treats all SNPs of each quality class as having equivalent qualities. The algorithm first focuses only on the SNPs of the highest quality. It creates an initial path of SNPs by walking in steps as close as possible but not exceeding a user-defined optimal spacing *s* (e.g. three kilobases). The algorithm then iteratively makes the best possible swap of each picked SNP with an adjacent unpicked SNP until no further improvement in the quality-dominated objective function can be made. If, after this first phase of the algorithm, there are any gaps in coverage grater than *s*, the algorithm attempts to fill these gaps with SNPs from the second best quality class using the same strategy. This process continues until there are no gaps greater than *s*, or all quality classes have been examined.

If all of the qualities are identical then the heuristic will produce the optimal solution for the fixed number of SNPs dictated by *s*. In this special case, the entire exact Pareto-optimal set can be generated by running the heuristic successively for each possible number of selected SNPs.

Exhaustive Enumeration Benchmark

The MAGMA parameters were: Version – Constrained substitution/deletion operators; Population Size – 100 (50 as archive); Seed with heuristic – NO; SNP coverage radius – 6000; init-low-genome-size – 2; init-high-genome-size – 5; species.crossover-prob – 0.7; species.mutation-prob – 0.13; species.substitution-prob – 0.7; species.insertion-prob – 0.08; species.deletion-prob – 0.22; coverage-score-sampling-dist – 1.

Analytically Seeded Benchmark

The MAGMA parameters were: Version – Constrained substitution/deletion operators; Population Size – 500 (250 as archive); Seed with heuristic – NO; SNP coverage radius – 3000; init-low-genome-size – 200; init-high-genome-size – 1250; species.crossover-prob – 0.8; species.mutation-prob – 0.13; species.substitution-prob – 0.5; species.insertion-prob – 0.1 species.deletion-prob – 0.4; coverage-score-sampling-dist – 500.

Hardware and Software

The software environment we employed was Java JRE: Classic VM (build 1.4.0, J2RE 1.4.0 IBM build cxia32140-20020917a (JIT enabled: jitc)). Our benchmark tests were run on a dual 500 MHz Celeron with 512 MB of RAM.

Glossary

Fitness: A non-negative real number. Solutions with lower fitness preferentially survive to the next generation. Fitness is dependent not only on the solution, but also on the current population of solutions.

Strength: The number of solutions dominated by a solution.

Pareto-optimal front: The image of the Pareto-optimal set in the objective space.

Optimization space: The space defined by the objective functions.

Solution space: The set of all possible solutions to a problem.

Genome: An encoding of a solution. There is a one-to-one correspondence between genomes and solutions, although some genomes and their associated solution may violate constraints. This evolutionary-algorithm definition is somewhat analogous to the biological definition of a genome.

Pareto optimality: The property of solution of having no other solution which is at least equal in all objective functions and greater in at least one objective function.

Pareto-optimal set: The set of all Pareto-optimal solutions.

Dominance: The property of being at least equal in all objectives, and better in at least one objective.

Abbreviations

SNP: single-nucleotide polymorphism

MAGMA: Multiobjective Analyzer for Genetic Marker Selection

SPEA: Strength-Pareto Evolutionary Algorithm

SPEA2: Strength-Pareto Evolutionary Algorithm 2

Authors' contributions

RH implemented the MAGMA algorithm in Java. EZ designed the evolutionary algorithm. JR conceived of the study, participated in its design and coordination, and drafted the manuscript. All authors contributed to

improvement of the algorithm, and read and approved the final manuscript.

Acknowledgements

Funding was provided in part by the Juvenile Diabetes Research Foundation Center for Bioinformatics. Chris Carlson helpfully broadened the Discussion.

References

1. Carlson CS, Newman TL and Nickerson DA: **SNPing in the human genome** *Curr Opin Chem Biol* 2001, **5(1):**78-85.
2. Schifreen RS, Storts DR and Buller AM: **The challenge of using SNPs in the understanding and treatment of disease** *Biotechniques* 2002, **Suppl:**14-6. 18, 20–1
3. Weiner MP and Hudson TJ: **Introduction to SNPs: discovery of markers for disease** *Biotechniques* 2002, **Suppl:**4-7. 10, 12–3
4. De La Vega FM, Dailey D, Ziegle J, Williams J, Madden D and Gilbert DA: **New generation pharmacogenomic tools: a SNP linkage disequilibrium Map, validated SNP assay resource, and high-throughput instrumentation system for large-scale genetic studies** *Biotechniques* 2002, **Suppl:**48-50. 52, 54
5. Heil J, Glanowski S, Scott J, Winn-Deen E, McMullen I, Wu L, Gire C and Sprague A: **An automated computer system to support ultra high throughput SNP genotyping** *Pac Symp Biocomput* 2002:30-40.
6. Gabriel SB, Schaffner SF, Nguyen H, Moore JM, Roy J, Blumenstiel B, Higgins J, DeFelice M, Lochner A, Faggart M, Liu-Cordero SN, Rotimi C, Adeyemo A, Cooper R, Ward R, Lander ES, Daly MJ and Altshuler D: **The structure of haplotype blocks in the human genome** *Science* 2002, **296(5576):**2225-9.
7. Jeffreys AJ, Ritchie A and Neumann R: **High resolution analysis of haplotype diversity and meiotic crossover in the human TAP2 recombination hotspot** *Hum Mol Genet* 2000, **9(5):**725-33.
8. Sherry ST, Ward MH, Kholodov M, Baker J, Phan L, Smigielski EM and Sirotkin K: **dbSNP: the NCBI database of genetic variation** *Nucleic Acids Res* 2001, **29(1):**308-11.
9. Fredman D, Siegfried M, Yuan YP, Bork P, Lehvaslaiho H and Brookes AJ: **HGVbase: a human sequence variation database emphasizing data quality and a broad spectrum of data sources** *Nucleic Acids Res* 2002, **30(1):**387-91.
10. Hubley R, Zitzler E, Siegel A and Roach J: **Multiobjective Genetic Marker Selection** In *Advances in Nature-Inspired Computation: the PPSN VII Workshops* Edited by: *David Corne. PEDAL, Reading, UK*; 2002:32-33.
11. Stram DO, Haiman CA, Hirschhorn JN, Altshuler D, Kolonel LN, Henderson BE and Pike MC: **Choosing haplotype-tagging SNPs based on unphased genotype data from a preliminary sample of unrelated subjects with an example from the Multiethnic Cohort Study** *Human Heredity* 2003.
12. Meng Z, Zaykin DV, Xu C-F, Wagner M and Ehm MG: **Selecting SNPs for Association Analyses** *Am J Hum Genet* 2003, **73:**115-130.
13. Zhang K, Deng M, Chen T, Waterman MS and Sun F: **A dynamic programming algorithm for haplotype block partitioning** *PNAS* 2002, **99:**7335.
14. Brusic V, Rudy G, Honeyman MC, Hammer J and Harrison LC: **Prediction of MHC class-II binding peptides using an evolutionary algorithm and artificial neural network** *Bioinformatics* 1998, **14(2):**121-130.
15. Zitzler E, Laumanns M and Thiele L: **SPEA2: Improving the Performance of the Strength Pareto Evolutionary Algorithm. Technical Report 103** *Zurich, Computer Engineering and Communication Networks Lab, Swiss Federal Institute of Technology* 2001.
16. Zitzler E, Laumanns M and Thiele L: **SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization** In *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems. Proceedings of the EUROGEN2001 Conference* Edited by: *Giannakoglou KC. Barcelona, Spain, International Center for Numerical Methods in Engineering*; 2002:95-100.
17. Zitzler E: **Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications** *Aachen, Shaker Verlag* 1999.
18. Luke S: **An Evolutionary Computation and Genetic Programming System** 2001 [http://www.cs.umd.edu/projects/plus/ec/ecj].
19. Luke S: **ECJ 8: A Java-based Evolutionary Computation and Genetic Programming Research System** 2001 [http://www.cs.umd.edu/projects/plus/ec/ecj].
20. Coello CAC: **A comprehensive survey of evolutionary-based multiobjective optimization** *Knowledge and Information Systems* 1999, **1(3):**269-308.
21. Coello CAC, Van Veldhuizen DA and Lamont G: **Evolutionary algorithms for solving multi-objective problems** *New York, Kluwer* 2002.
22. Deb K: **Multi-objective genetic algorithms: Problem difficulties and construction of test functions. Technical Report No. CI-49/98** *Department of Computer Science/XI, University of Dortmund, Germany* 1998.
23. Deb K: **Multi-objective optimization using evolutionary algorithms** *Chichester, UK, Wiley* 2001.
24. Fonseca CM and Fleming PJ: **An overview of evolutionary algorithms in multiobjective optimization** *Evolutionary Computation* 1995, **3(1):**1-16.
25. Horn J: **F1.9 multicriteria decsion making** In: *Handbook of Evolutionary Computation* Edited by: *Bäck T, Fogel DB, Michalewicz Z. Bristol, UK, Institute of Physics Publishing*; 1997.
26. Laumanns M, Zitzler E and Thiele L: **On The Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-Objective Optimization** In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)* Edited by: *Zitzler E. Berlin, Germany, Springer-Verlag*; 2001:181-196.