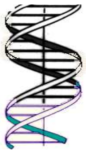
Solving the Inverse Kinematics Problem With Genetic Algorithms

Leon James Derman



Abstract

This paper describes a method of utilizing genetic algorithms to search for the solution of an inverse kinematics problem in which there is a high degree of kinematic redundancy. A genetic algorithm is a search technique that attempts to simultaneously satisfy all system constraints while minimizing a given objective function. Various reordering operators are introduced which help the genetic algorithm effectively solve problems (such as this one) where changing one gene of an individual has an effect on the expression of one or more of the individual's other genes. This gene interdependance is known as epistatis. Problems which contain a high degree of epistasis have been shown to be difficult to solve using genetic algorithms. In this case, genetic algorithms are applied using a simple encoding of the inverse kinematic problem into genetic algorithm space. Genetic algorithms with various traits are compared. Finally, the best performing genetic algorithm is compared to a reasonably well performing random search (applied to the same problem).

1 INTRODUCTION

For this project, we will attempt to use genetic algorithms to solve an inverse kinematics problem which contains a high level of redundancy. The inverse kinematics problem, which is typically applied to a model which is composed of segments and joints, consists of determining what local rotational values need to be applied to one or more of the joints in order to move the endpoint of the last segment from one given world coordinate to another given world coordinate.

The inverse kinematics problem that I shall be addressing shall be essentially determining appropriate rotation angles for each joint specified (interactively) in Professor Wilhelms fauna system [10]. However, I shall be enforcing joint limits.

I believe that genetic algorithms are well suited to solve the inverse kinematics problem for a number of important reasons. First of all, most gradient search techniques are susceptible to becoming trapped at a local minimum. Genetic algorithms are must less susceptible to this problem because each iteration considers a population of possible steps, instead of a single step. This may sound costly, but since the genetic operations are quick, the search is efficient providing the function used to calculate the fitness of each individual is efficient. Secondly, the basic genetic algorithm operators, crossover and mutation, are blind to the problem being solved. Hence, genetic algorithms are widely applicable to problems in general, so it is easy to include additional constraints and/or dynamically alter the environment in radical ways and still have the search converge on the goal state; i.e., genetic algorithms are robust.

A genetic algorithm consists of some number of genetic operators, (such as crossover, mutation, etc. - described later), a population (group) of individuals to which the genetic operators are applied, and an evaluation function which determines the fitness of each individual in the current environment. Each individual contains an instance of a possible solution to the problem. The problem being solved is encoded into a bit string. This bit string (or some function of it) must be capable of representing all desired solutions, since the genetic operators merely flip bits and copy/move portions of bit strings - genetic operators do not expand or reduce the size of an individuals bit string.

A genetic algorithm coverges on the goal state in the following manner. A population of possible "solutions" to the search problem is created using some form of a heuristic. In general, the closer at least some individuals (or combinations of individuals) in the population are to the goal state, the quicker the search will converge. This is evident when we consider that the majority of genetic algorithm search consists of moving about various portions of bit strings in an attempt to find the solution. So if the bit strings we start out with are not at least somewhat representative of the encoding of the solution, then we must rely on the mutation operator to mutate a sufficient number of these bit strings so that we have material for the crossover operator to work with. For each iteration of the search, the fitness of every individual in the population is calculated. Individuals are selected probabilistically, with those having a higer fitness being more likely to be selected. Each time two individuals are selected, there is a chance that they shall either live on into the next iteration of the search unmodified, or that they will be modified by crossover and/or mutation. When crossover is applied to two individuals, a bit position is randomly chosen as the "cut site" and the bit data from the this point on is swapped between the two individuals. Mutation is also randomly applied to the bits which compose an individuals genetic encoding. Mutation merely flips the bit value of the bit being mutated. The probability that crossover and mutation occur are parameters to the genetic algorithm. In general, a crossover probability of 60Analogous to organisms in real life, the genes of individuals in genetic search may be either haploid, meaning that the individual has one copy of each gene, or diploid, meaning that the individual has two copies of each gene. When the population consists of individuals with diploid chromosomes, some type of dominance rule must be applied to the individual's genotype in order to determine the corresponding phenotype (expressed traits) of that individual.

There are some fundamental properties of genetic algorithms that force one to introduce some genetic operators other than the basic operators crossover and mutation, when applying this search technique to the inverse kinematics problem. As mentioned previously, certain codings of the inverse kinematics problem contain a very high degree of epistasis - meaning that changing the value of one gene can have a great influence on the fitness of many of the individual's other genes. For example, if we encode the state space for the an inverse kinematics problem in which we are to control 5 joints, as one chromosome which contains, say 30 bits (10 bits for each x/y/z rotation which allows each joint rotation to range from -360 to 360) for each joint, this yields a chromosome with 150 bits. Clearly, with this encoding scheme, we have a case where rotating a joint (changing bits for one joint) affects the usefulness of the current values of all joints distal to it. Several techniques shall be presented in this paper which attempt to overcome this problem of epistasis.

2 RELATED WORK

In 1986, Chang [1] proposed a closed-form solution to the ik problem with kinematic redundancy using the Lagrangian multiplier method. Like all local search methods, the effectiveness of Lagrangian multipliers can be greatly diminished when this search algorithm is applied to problems with strongly coupled constraints. Furthermore, the optima reached is a local optima which may be worse then the global optima. Of course, we are not guarenteed to find the global optima when using genetic algorithms.

When applying a global search method to the inverse kinematics problem, we are immediately confronted with the problem of finding a reasonable way of constraining how we modify joint angles in an attempt to shrink the search space of the problem to a manageable size. Since genetic algorithms are blind to the details of the actual problem, it is mainly the properties of the genetic operators that determine exactly how the search progresses towards the goal. Of course, the objective function used to evaluate encodings plays a major role in the search path, as well, since it provides the feedback to the genetic algorithm regarding the fitness of its individuals.

Parker [4] presents a paper which serves a good introduction to genetic algorithms, but fails to address any of the problems encountered when genetic algorithms are applied to more complex problems, like the problem that Buckley and his colleagues attempted to solve [6].

One interesting application of genetic algorithms to the inverse kinematics problem is presented by Chocron [8] where a two-level genetic algorithm is utilized. Chocron and Bidaud break their problem down in such a way as to facilitate an encoding in genetic algorithm space which is readily tractable for a simple genetic algorithm. Moreover, their objective function includes the notion of joint dexterity [11], and other useful evaluation criteria. Others have attempted to use genetic alogorithms to evolve connections in the brains of simulated organisms [9]. However, in this paper, Sims is using genetic algorithms to evolve control parameters for the neural network "brains" of individuals, which puts the burden of learning movement on the neural networks as opposed to the genetic algorithm. The use of genetic algorithms to evolve control parameters, is intriguing, since it allows one to (possibly) decouple the genetic algorithm from any particular difficulties that may exist in the encoding of the problem in genetic algorithm space - i.e. we have another level of abstraction. Furthermore, we are quite possibly reducing the size of an individual's genotype since we need only encode the control parameters, as opposed to encoding a huge number of variables that the control parameters influence.

Before I discuss how I intend to approach this problem let me first cite a failure to achieve almost exactly what I am attempting to do and then explain what I am going to do differently. In the paper by K. Buckley, S. Hopkins and B. Turton [6], Buckley and his colleagues attempt to use genetic algorithms to solve an inverse kinematics problem in which they are controlling a "highly kinematically redundant manipulator," (one with a high kinematic redundancy).

Their results, even after attempts to "enhance" their genetic algorithm, are "not entirely successful." They conclude that this is due to the fact that the inverse kinematics problem they are attempting to solve has a high degree of epistasis. In genetic terms this means that the alleles at one locus have an effect on the expression of alleles at another locus. What this means in term of the inverse kinematics problem is that moving a joint has an effect on all joints distal to it. Genetic algorithms which do not include any reordering operations tend to perform worse than random search when they are applied to problems with a high degree of epistasis [3]. However the authors's attempts to "enhance" their basic genetic algorithm do not directly address this problem of epistasis, even though there are methods which have been shown to improve genetic algorithm effectiveness at solving problems with a high degree of epistasis - methods like inversion and other reordering operations [2].

3 EVOLVING A STRATEGY

The first thing to decide upon when attempting to apply a genetic algorithm to a given problem is how to encode the state space of the problem into a bit string. For this problem, we purposely decided upon the "naive" encoding of assigning ten bits per degree of freedom / per joint. So, the genetic encoding directly contains the x/y/z axis rotation values for each joint required to move the end effector from the start position to the goal position. We chose nine bits to represent a given joint's rotation in either x y or z because this size of a bit string allows rotational values to range from -360 - 360 inclusive. Each joint has three of these nine bit strings, one for x, y, and z rotation. We chose to use diploid individuals, because during initial tests of populations composed on haploid individuals, the population did not perform well. In this encoding, each gene is analogous to the x/y/z rotation values for a joint. As mentioned previously, each gene is further composed of three alleles, where the first allele is for x rotation, the second for y, etc. Because the population is composed of diploid individuals, some form of dominance is required so that we know how the two alleles for a given trait are expressed. For example, blue eyes are a recessive trait, and hence a "dominated" trait - if an individual has genes for both blue eyes and brown eyes, the brown eyes are expressed in the individual but the individual's offspring may have blue eyes. It has been hypothesized that bestowing populations with diploidance in genetic search is a good thing because traits that were once well suited for the environment are preserved in the population even if they are not currently fit (as recessive genes) [3]. The fitness function is simply the distance from the current position of the end effector to the goal position.

4 RESULTS

Initial tests were performed utilizing a simple genetic algorithm with a haploid chromosome representation on a twelve segment tail. On average, the population took several hundred iterations to converge. Clearly, these results were not encouraging. Performance when using a diploid chromosome representation on the same twelve segment tail increased somewhat, but the system still did not always converge. After extensive testing, it was clear that the diversity of the system's population quickly degraded. The shining stars of fitness quickly crowded out the rest of the population, with the net effect being that the crossover operator found itself with insufficient diversity to perform its job well. To combat this situation we chose to introduce "regularization steps" in which a simple but effective heuristic was employed to "fine-tune"