Proceedings of the 11th World Congress in Mechanism and Machine Science
August 18–21, 2003, Tianjin, China
China Machinery Press, edited by Tian Huang

Evolutionary Algorithm for Robot Task Space Optimization

Joshua D. Petitt and Karol Miller
School of Mechanical Engineering, The University of Western Australia, Perth 6009, Australia
e-mail:petitj01@mech.uwa.edu.au

Abstract: This paper describes the application of an evolutionary algorithm for optimization of robot physical parameters. Preliminary tests were conducted with the algorithm to determine the optimum set of algorithm parameters. The kinematics of laproscopic surgery are discussed and used to generate a set of operating points called a *task space*. Optimal sets of link lengths are found using the sample mean and standard deviation of each of the actuator values as a performance rating.

Keywords: Evolutionary algorithm, task space

1 Introduction

In this paper, a specific implementation of an evolutionary algorithm, programmed in MATLAB, is used to optimize the physical parameters for a surgical assist manipulator (SAM). The use of different parameters for the evolutionary algorithm will be discussed and the optimal set of parameters, for this implementation, is given.

2 Evolutionary Algorithm

2.1 Description

Neo-Darwinism states that four processes act upon a species: reproduction, mutation, competition, and selection [1]. These processes can be expressed, algorithmically, as operations on members of the population, or a finite set.

$$\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \tag{1}$$

Each member in $\boldsymbol{X}$ can be described by a set of attributes

$$\boldsymbol{x}_i = [a_1, a_2, \ldots, a_m] \tag{2}$$

which can be evaluated by a *fitness function*, a linear or non-linear function of the attribute vector, that usually evaluates to a scalar quantity, or the *performance* of the population member.

$$p_i = \mathrm{F}(\boldsymbol{x}_i) \tag{3}$$

The general view on the effect of natural evolution is the continuing improvement of population fitness in a dynamic landscape (i.e. the fitness function is changing). This is in contrast to most engineering optimization problems where the fitness function is static throughout the computation of the algorithm and the goal of the optimization is to find the *best* solution, or the set of attributes that minimize or maximize the value of the fitness function. Note that the choice of maximizing or minimizing is dependant on the nature of the fitness function.

The algorithm presented has the following properties

- Uniform random distribution of initial population
- Uniform random selection of tournament opponents
- Tournament selection of dominant parent
- Roulette selection of passive parent
- Crossover follows uniform distribution
- Mutation follows normal distribution
- Elitist approach for selection of next generation

Uniform random distribution of the initial population over the entire state space is important to ensure that at least one individual is close to the maximum value of the performance function.

Competition among members of a species, especially among male members, is a fundamental mechanism by which the weak members of the population are culled. The results of these competitions are usually not death, but result in little chance to mate and produce offspring [1]. The algorithm presented here, although does not include the aspect of sexuality of specific members, does incorporate competition among members. To choose a dominant parent, two distinct members are randomly chosen from the population with uniform likelihood of being chosen. The member with the higher performance is chosen as the dominant parent.

The choice of passive parent is either by uniform random selection or the *roulette wheel approach*, where the fitness of the entire population is summed and denoted as *S*. A uniform random variable, *s*, is generated over [0,*S*]. The population member whose fitness plus the fitness of all lower population members is greater or equal to *s* is chosen as the passive parent. This approach is described in [1].

During the mating portion of the algorithm, the attributes of each parent are combined to create the child attributes. The crossover operation is achieved by

$$a = \alpha \cdot a_1 + (1-\alpha) \cdot a_2 \tag{4}$$

where α is a uniform random variable between zero and unity. Because the evolutionary algorithm works with the phenotype, each attribute only influences the corresponding attribute of the child. This approach is easily implemented and is desirable because the result is a real number guaranteed to lie between a_1 and a_2.

After crossover of the attributes, a mutation of each attribute is introduced. The mutation is a random variable from a normal distribution with mean zero and a small value for the standard deviation. It is interesting to note at this point, that the advantages of using crossover and mutation, or exclusively one of the operations is shown in the results of this experiment. It was determined that a mutation parameter is necessary for the algorithm to find solutions on the extreme edges of the search space.

After each child member is created, it is placed into a new population. The dominant and passive parents may also pass onto the new population according to the *persistence* of the algorithm. If the persistence is zero, then neither parent passes on, if the persistence is unity, then the dominant parent is allowed to join the new population. If the persistence is two, then both parents join the new population. The persistence parameter of the algorithm is important because it allows the population to retain members with high performance values.

Finally, the new population is ordered according to performance of the members and the top N_p members are kept, and the others discarded, where N_p is the number of population members. This operation is called *elitist* selection and is important to keep population size constant during algorithm execution.

2.2 Benchmark Testing

An evolutionary algorithm has many parameters, which may or may not affect the overall performance of the algorithm. Therefore, before conducting an optimization of a real system, the algorithm was tested on four test performance surfaces to determine the best combination of parameters. The following four functions that were tested are shown with the corresponding surface plots.

$$z = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7 \qquad (5)$$

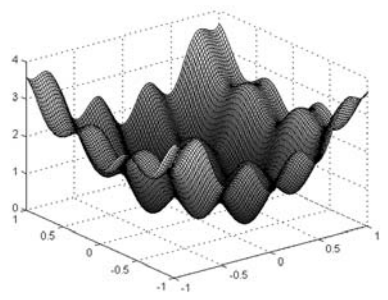


Figure 1 – Test Performance Surface

$$z = x^2 + 2y^2 - 0.3\cos(3\pi x)\cos(4\pi y) + 0.3 \qquad (6)$$

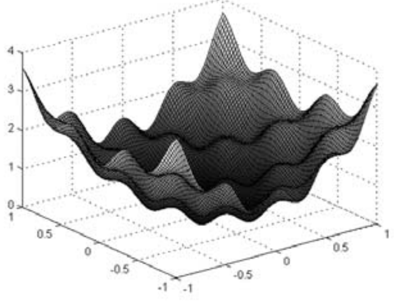


Figure 2 – Test Performance Surface

$$z = x^2 + 2y^2 - 0.3\cos(3\pi x + 4\pi y) + 0.3 \qquad (7)$$

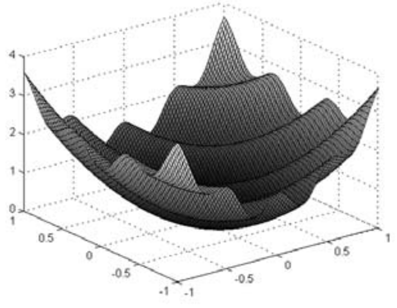


Figure 3 – Test Performance Surface

$$z = x + y + 2 \qquad (8)$$

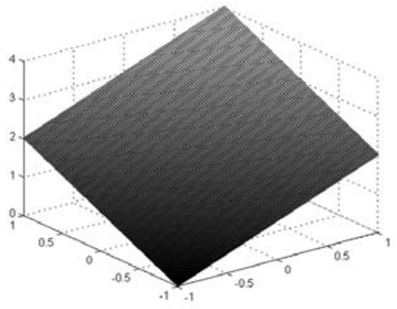


Figure 4 – Test Performance Surface

Notice that each function is a function of two variables, x and y, and each has one unique global minimum in the range of [-1,1]. The global minimum for the first three functions is located at [0,0], but the minimum value for the fourth function is at [-1,-1]. This is an important case because it will test the ability of the algorithm to find a minimum that lies at a boundary.

To test the effects of parameter variation, a set of parameters was chosen, then the values of x and y were 'optimized' for each function 100 times and the sample mean and standard deviation of the performances were calculated. For an optimization algorithm, one would desire the return value to be consistent, if not the same, each time the algorithm is executed. Obviously, one would also desire the algorithm to return the global minimum, or a close approximation, for the function. This leads to two null independent hypotheses for each test set.

The mean return value for parameter set A is less than the mean return value for parameter set B.

The standard deviation of the return value for parameter set A is less than standard deviation of the return value for parameter set B.

Table 1 shows the parameter sets that were tested. The parameters were varied according to each row in the table, and the effects tested on each of the four test functions.

The effects of three other parameters were also observed for each parameter set; these are population size, number of epochs and random mutation. The number of *epochs* is equivalent to the number of generations, or iterations, for which the algorithm calculates.

Table 1 – Optimization Test Parameter Sets

Competition	Selection	Elitism	Persistence
1	Random	No	0
2	Random	No	0
2	Random	No	1
2	Random	Yes	1
2	Roulette	Yes	1
2	Roulette	Yes	2
3	Roulette	Yes	2
2	Roulette	No	1

During an actual optimization problem, the evaluation time of the performance function is large compared to the execution time of the algorithm. Therefore, when comparing the effects of population size versus number of epochs it is important to keep the number of performance evaluations equal. In this specific implementation of the algorithm, the performance of each member of the population is only evaluated once, when the member is created. Because the member's attributes are not modified during algorithm execution, the performance of the member is constant. For every test, a pair of parents produced only one child. Thus, the number of performance evaluations can be calculated as

$$N = population \times (1 + epochs \times children) \tag{9}$$

2.3 Benchmark Test Results

The optimal set of algorithm parameters were

Competition	Selection	Elitism	Persistence
2	**Roulette**	**Yes**	**1**

Figures 5 and 6 show the mean performance value for the optimal set of parameters when optimizing equation (2). Equations (1) and (3) exhibit the same characteristics, probably because the minimum value of the function occurs within the search boundaries.

Figure 8 illustrates the mean performance value for the optimal set of parameters when optimizing equation (4). Notice that this graph shows a heavy dependence on the mutation rate in order to find the optimal solution.

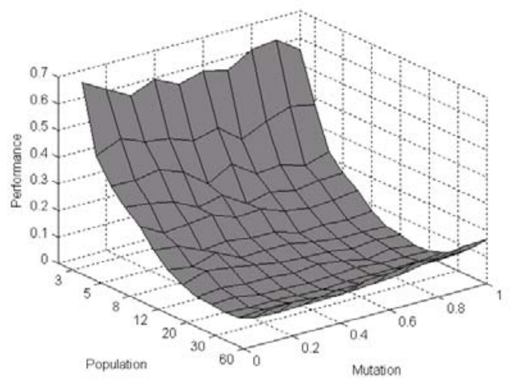


Figure 5 – Mean performance versus population size and mutation rate for equation (2) with optimum parameters

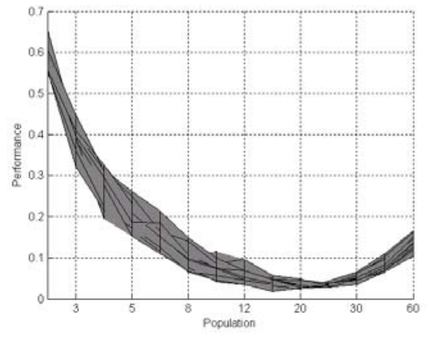


Figure 6 – Mean performance vs. population size for equation (2) with optimal parameters

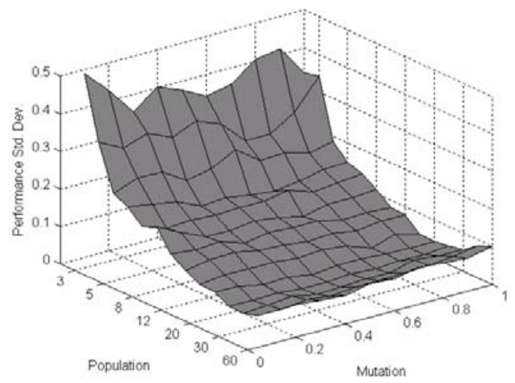


Figure 7 – Standard deviation of performance for equation (2) with optimal parameters

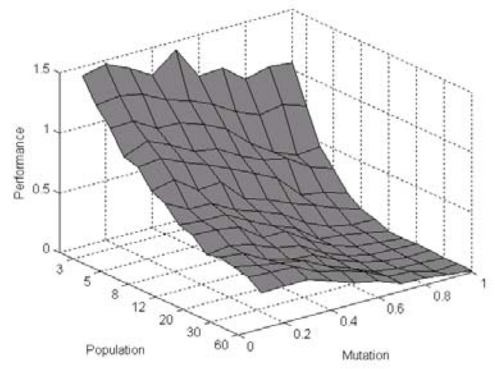


Figure 8 – Mean performance versus population size and mutation rate for equation (4) with optimum parameters

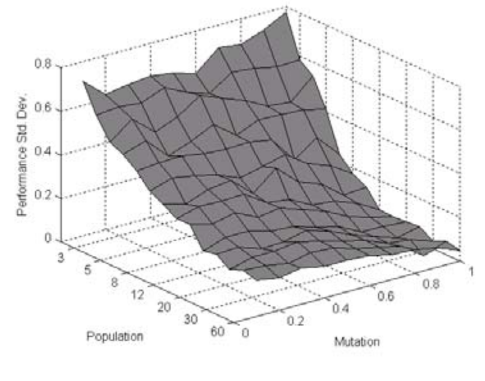


Figure 9 – Standard deviation of performance for equation (4) with optimum parameters

3 Optimal Design

Many researchers have examined optimal design of mechanisms. Usually the approach taken is the minimization of an objective, or cost function, using linear or nonlinear optimization techniques. Miller and Stock [2][3] use a combinatorial approach for optimizing the workspace volume of the DELTA and Linear DELTA style parallel robots, respectively. The performance of the workspace is quantified in the first case as the total working volume of the manipulator. In the second case, the inverse of the condition number of the Jacobian matrix for the manipulator kinematics is used to quantify the manipulator performance.

The drawback to the approach taken above is the accuracy of the solution versus the time taken to compute the performance.

3.1 Task-Driven Optimization

The workspace of a generic manipulator has often been the prime target of optimization techniques, to which there are many approaches [2][3]. However, these procedures are either analytical or take a long to compute because of combinatorial explosion. Therefore, an alternative method will be proposed here called "task-driven optimization". The focus behind task driven optimization is not to create the entire workspace of the manipulator, but to only analyze points in the vicinity of a critical operating point, or set of operating points. This allows for more points to be evaluated in the vicinity of the precision location(s).

This approach was taken by [4] in optimizing the link lengths of a planar four-bar linkage for traveling through precision points. In the six-dimensional case, it is assumed that the approach vector of the tool is directly toward the center of the precision area. Thus, only four search dimensions are needed, three for position and one for the final orientation.

3.2 Laproscopic Surgery

Laproscopic surgery, also called 'keyhole' surgery, is a surgical procedure whereby the surgeon operates on the patient through a set of small incisions, typically three total, one for the laproscope and two for operating ports. The tools that are used for the operation are long (~30 cm), slender devices with specialized tips for specific functions. A laproscope, a long, slender viewing device is also inserted into the patient to allow the surgeon to see the operation area. Advanced imaging techniques, such as magnetic resonance imaging, can replace the need for the laproscope, but the operation is still performed with similar tools. The unwieldy nature of the tools makes the job of the surgeon difficult and results in a lack of dexterity and sensory feedback, which is the main drawback to the procedure, from the surgeon's perspective [5][6]. The insertion of the tool through the incision also reduces the degrees of freedom of the tool tip from six to four.

To overcome the problem of positioning the surgical instruments, the introduction of a robotic manipulator into the operating theatre has been a proposed solution. Chenzei and Miller [7] have developed a five degree-of-freedom SAM that is compatible with a magnetic resonance environment. Other researchers have also developed designs for SAMs [8][9][10], based on serial, hybrid and parallel structures.

3.2.1 Manipulator kinematics

The manipulator that is optimized is a six degree-of-freedom fully parallel manipulator. The actuators are prismatic and are joined to the connecting links by spherical joints. The traveling plate is attached to the connecting links by universal joints. Figure 10 is a diagram of the design. The manipulator connecting links are the focus of the optimization, and the optimal set of lengths for a given task is desired.

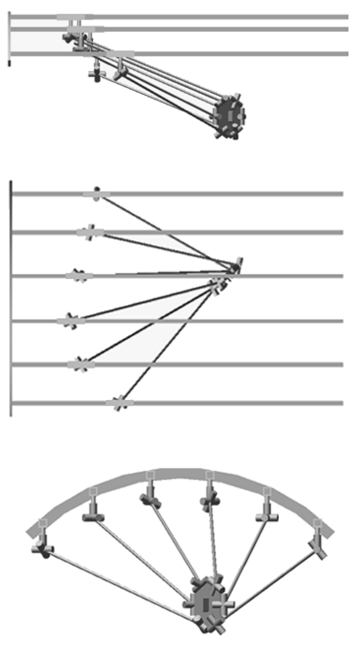
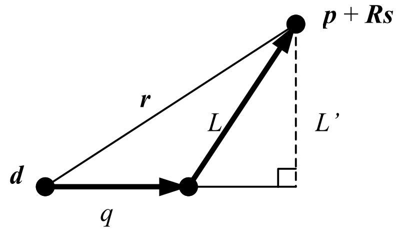


Figure 10 – Diagram of a six DOF parallel linear robot

The inverse kinematic constraint equations can be derived as follows: Let the traveling plate coordinate system be defined by a point $\boldsymbol{p}$ and a [3x3] rotation matrix, $\boldsymbol{R}$. Let $\boldsymbol{s}$ represent a displacement from $\boldsymbol{p}$ to the connection point between the connecting link and the traveling plate. Now let each actuator be represented by a point $\boldsymbol{d}$ and a direction vector $\boldsymbol{u}$. Let the distance $\boldsymbol{d}$ to the traveling plate connection point be denoted by $\boldsymbol{r}$, which can be calculated by equation (10). Knowing $\boldsymbol{r}$ allows L' to derived, thus the distance, q, along $\boldsymbol{u}$ which the connect link lies.



$$\boldsymbol{r}_i = \boldsymbol{p} + \boldsymbol{R}\boldsymbol{s}_i - \boldsymbol{d}_i \tag{10}$$

$$L'_i = \sqrt{\boldsymbol{r}_i \bullet \boldsymbol{r}_i - (\boldsymbol{r}_i \bullet \boldsymbol{u}_i)^2} \tag{11}$$

$$q_i = \boldsymbol{r}_i \bullet \boldsymbol{u}_i \pm \sqrt{L_i^2 + (\boldsymbol{r}_i \bullet \boldsymbol{u}_i)^2 - \boldsymbol{r}_i \bullet \boldsymbol{r}_i} \tag{12}$$

Note that the inverse kinematic equation yields two solutions for the actuator values. These solutions correspond to the two assembly modes for the manipulator.

3.2.2 Performance function

In constructing a performance function for the manipulator, two considerations were made. First, it is desirable that the physical actuators are as small as possible, thus the stroke length of the actuator should be minimized. Second, in order to minimize the length of the connecting links, the actuators should operate close to the maximum value. Therefore, using these two criteria, the following performance function is defined

$$p = \sum_{i=1}^{6} (\overline{X_i} - S_i^2) \tag{12}$$

where $\overline{X_i}$ is the sample mean value and S_i^2 is the sample variance of the *i*th actuator values during task execution.

Conclusions

Figure 11 shows the optimized plots of the actuator values for the laproscopic task space. To optimize the actuator lengths, the algorithm was set to use 120 points to define the precision location, a population size of 30 members, 4 epochs, and mutation parameter of 0.5 and the algorithm attempted to maximize the value of the performance. One will note that the maximum value of each of the links is approximately 1, which is the maximum travel of the actuator.

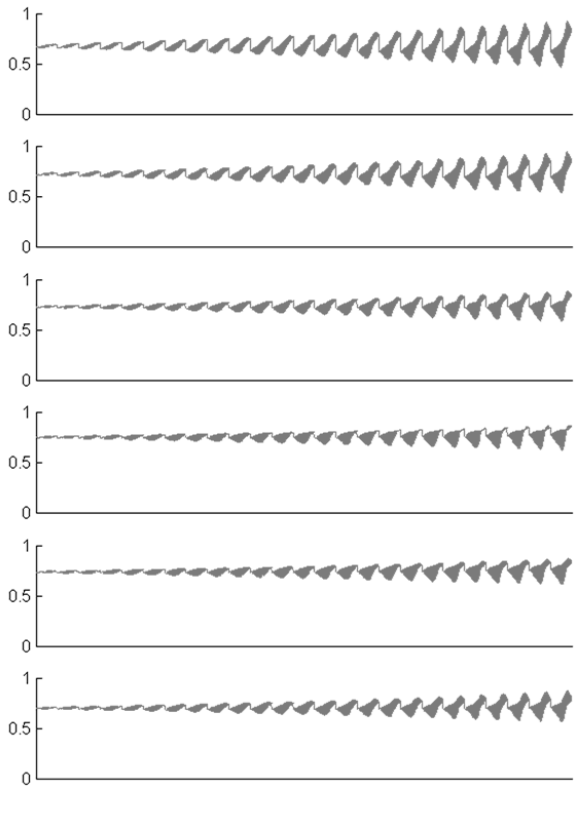


Figure 11 – Actuator output values for the laproscopic task space

The algorithm took approximately 19 seconds to calculate the optimal length lengths for the task. This is clearly an acceptable execution time, which promotes an iterative design process. Using an optimization technique based upon the workspace, depending on the accuracy, can take hours or days because of the sheer number of points that are evaluated. This limits the designer in flexibility because the majority of the time is spent on computation and not on design.

Acknowledgment

The author would like to thank Dr. Kiyoyuki Chenzei and the members of the Surgical Planning Laboratory at Brigham and Women's Hospital in Boston, Mass. USA, for the invaluable information concerning MRI surgery.

I would also like to thank Stéphane Muguet for providing the illustration of the six degree-of-freedom robot. Finally, thank you to Kevin Hayward of the School of Mechanical Engineering at The University of Western Australia for the stimulating discussion concerning evolutionary algorithms.

References

[1] Fogel, D. B. (1995). Evolutionary Computation. New York, The Institute of Electrical and Electronic Engineers, Inc.

[2] Miller, K. (2002). "Maximization of Workspace Volume of 3-DOF Spatial Parallel Manipulators." ASME Journal of Mechanical Design **124**: 1-4.

[3] Stock, K., Miller, K. (2002). "Optimal Kinematic Design of Spatial Parallel Manipulators: Application to Linear Delta Robot." ASME Journal of Mechanical Design. accepted July 2002

[4] Roston, G. P. and R. H. Sturges (1996). "Genetic algorithm synthesis of four-bar mechanisms." Artificial Intelligence for Engineering Design, Analysis and Manufacturing **10**: 371-390.

[5] Treat, M. R. (1996). A Surgeon's Perspective on the Difficulties of Laparoscopic Surgery. Computer-Integrated Surgery. R. H. Taylor, S. Lavallée, G. C. Burdea and R. Mösges. Cambridge, The MIT Press: 736.

[6] Tendrick, F., R. W. Jennings, et al. (1996). Perception and Manipulation Problems in Endoscopic Surgery. Computer-Integrated Surgery. R. H. Taylor, S. Lavallée, G. C. Burdea and R. Mösges. Cambridge, The MIT Press: 736.

[7] Chinzei, K. and K. Miller (2001). "Towards MRI Guided Surgical Manipulator." Med. Sci Monit. **7**(1): 153-163.

[8] Taylor, R. H., J. Funda, et al. (1996). A Telerobotic Assistant for Laparoscopic Surgery. Computer-Integrated Surgery. R. H. Taylor, S. Lavallée, G. C. Burdea and R. Mösges. Cambridge, The MIT Press: 736.

[9] Khodabandehloo, K., P. N. Brett, et al. (1996). Special-Purpose Actuators and Architectures for Surgery Robots. Computer-Integrated Surgery. R. H. Taylor, S. Lavallée, G. C. Burdea and R. Mösges. Cambridge, The MIT Press: 736.

[10] Grace, K. W. (1995). Kinematic Design of an Ophthalmic Surgery Robot and Feature Extracting Bilateral Manipulation. Mechanical Engineering. Evanston, Illinois, Northwestern University: 85.