UNIVERSITY OF ALBERTA

ROBOT PATH PLANNING USING A GENETIC ALGORITHM APPROACH

BY

© HONG HAO

A thesis submitted to the **Faculty of Graduate Studies and Research** in partial fulfilment of the requirements for the degree of **MASTER OF SCIENCE**.

DEPARTMENT OF MECHANICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1994

ISBN 0-315-95124-9

Name [illegible]

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

[illegible] — SUBJECT TERM

0542 — SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture	0729
Art History	0377
Cinema	0900
Dance	0378
Fine Arts	0357
Information Science	0723
Journalism	0391
Library Science	0399
Mass Communications	0708
Music	0413
Speech Communication	0459
Theater	0465

EDUCATION

General	0515
Administration	0514
Adult and Continuing	0516
Agricultural	0517
Art	0273
Bilingual and Multicultural	0282
Business	0688
Community College	0275
Curriculum and Instruction	0727
Early Childhood	0518
Elementary	0524
Finance	0277
Guidance and Counseling	0519
Health	0680
Higher	0745
History of	0520
Home Economics	0278
Industrial	0521
Language and Literature	0279
Mathematics	0280
Music	0522
Philosophy of	0998
Physical	0523
Psychology	0525
Reading	0535
Religious	0527
Sciences	0714
Secondary	0533
Social Sciences	0534
Sociology of	0340
Special	0529
Teacher Training	0530
Technology	0710
Tests and Measurements	0288
Vocational	0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language	
General	0679
Ancient	0289
Linguistics	0290
Modern	0291
Literature	
General	0401
Classical	0294
Comparative	0295
Medieval	0297
Modern	0298
African	0316
American	0591
Asian	0305
Canadian (English)	0352
Canadian (French)	0355
English	0593
Germanic	0311
Latin American	0312
Middle Eastern	0315
Romance	0313
Slavic and East European	0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy	0422
Religion	
General	0318
Biblical Studies	0321
Clergy	0319
History of	0320
Philosophy of	0322
Theology	0469

SOCIAL SCIENCES

American Studies	0323
Anthropology	
Archaeology	0324
Cultural	0326
Physical	0327
Business Administration	
General	0310
Accounting	0272
Banking	0770
Management	0454
Marketing	0338
Canadian Studies	0385
Economics	
General	0501
Agricultural	0503
Commerce-Business	0505
Finance	0508
History	0509
Labor	0510
Theory	0511
Folklore	0358
Geography	0366
Gerontology	0351
History	
General	0578
Ancient	0579
Medieval	0581
Modern	0582
Black	0328
African	0331
Asia, Australia and Oceania	0332
Canadian	0334
European	0335
Latin American	0336
Middle Eastern	0333
United States	0337
History of Science	0585
Law	0398
Political Science	
General	0615
International Law and Relations	0616
Public Administration	0617
Recreation	0814
Social Work	0452
Sociology	
General	0626
Criminology and Penology	0627
Demography	0938
Ethnic and Racial Studies	0631
Individual and Family Studies	0628
Industrial and Labor Relations	0629
Public and Social Welfare	0630
Social Structure and Development	0700
Theory and Methods	0344
Transportation	0709
Urban and Regional Planning	0999
Women's Studies	0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture	
General	0473
Agronomy	0285
Animal Culture and Nutrition	0475
Animal Pathology	0476
Food Science and Technology	0359
Forestry and Wildlife	0478
Plant Culture	0479
Plant Pathology	0480
Plant Physiology	0817
Range Management	0777
Wood Technology	0746
Biology	
General	0306
Anatomy	0287
Biostatistics	0308
Botany	0309
Cell	0379
Ecology	0329
Entomology	0353
Genetics	0369
Limnology	0793
Microbiology	0410
Molecular	0307
Neuroscience	0317
Oceanography	0416
Physiology	0433
Radiation	0821
Veterinary Science	0778
Zoology	0472
Biophysics	
General	0786
Medical	0760

EARTH SCIENCES

Biogeochemistry	0425
Geochemistry	0996
Geodesy	0370
Geology	0372
Geophysics	0373
Hydrology	0388
Mineralogy	0411
Paleobotany	0345
Paleoecology	0426
Paleontology	0418
Paleozoology	0985
Palynology	0427
Physical Geography	0368
Physical Oceanography	0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences	0768
Health Sciences	
General	0566
Audiology	0300
Chemotherapy	0992
Dentistry	0567
Education	0350
Hospital Management	0769
Human Development	0758
Immunology	0982
Medicine and Surgery	0564
Mental Health	0347
Nursing	0569
Nutrition	0570
Obstetrics and Gynecology	0380
Occupational Health and Therapy	0354
Ophthalmology	0381
Pathology	0571
Pharmacology	0419
Pharmacy	0572
Physical Therapy	0382
Public Health	0573
Radiology	0574
Recreation	0575
Speech Pathology	0460
Toxicology	0383
Home Economics	0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry	
General	0485
Agricultural	0749
Analytical	0486
Biochemistry	0487
Inorganic	0488
Nuclear	0738
Organic	0490
Pharmaceutical	0491
Physical	0494
Polymer	0495
Radiation	0754
Mathematics	0405
Physics	
General	0605
Acoustics	0986
Astronomy and Astrophysics	0606
Atmospheric Science	0608
Atomic	0748
Electronics and Electricity	0607
Elementary Particles and High Energy	0798
Fluid and Plasma	0759
Molecular	0609
Nuclear	0610
Optics	0752
Radiation	0756
Solid State	0611
Statistics	0463

Applied Sciences

Applied Mechanics	0346
Computer Science	0984
Engineering	
General	0537
Aerospace	0538
Agricultural	0539
Automotive	0540
Biomedical	0541
Chemical	0542
Civil	0543
Electronics and Electrical	0544
Heat and Thermodynamics	0348
Hydraulic	0545
Industrial	0546
Marine	0547
Materials Science	0794
Mechanical	0548
Metallurgy	0743
Mining	0551
Nuclear	0552
Packaging	0549
Petroleum	0765
Sanitary and Municipal	0554
System Science	0790
Geotechnology	0428
Operations Research	0796
Plastics Technology	0795
Textile Technology	0994

PSYCHOLOGY

General	0621
Behavioral	0384
Clinical	0622
Developmental	0620
Experimental	0623
Industrial	0624
Personality	0625
Physiological	0989
Psychobiology	0349
Psychometrics	0632
Social	0451

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **HONG HAO**

TITLE OF THESIS: **ROBOT PATH PLANNING USING A GENETIC ALGORITHM APPROACH**

DEGREE: **MASTER OF SCIENCE**

YEAR THIS DEGREE GRANTED: **FALL, 1994**

[illegible]

11, 10738-85 Ave

Edmonton, Alberta

T6E 2K8

Date: OCT. 6 .. 1994

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **ROBOT PATH PLANNING USING A GENETIC ALGORITHM(GA) APPROACH** submitted by **Hong Hao** in partial fulfilment of the requirements for the degree of **Master of Science.**

Roger W. Toogood
Dr. R.W. Toogood

A W Lipsett
Dr. A.W. Lipsett

[illegible]
Dr. R. G. Goebel

Date: [illegible] October 1994

ABSTRACT

This thesis describes the use of Genetic Algorithms(GAs) for finding a continuous collision-free path for a 3-DOF revolute robot manipulator between arbitrary start and goal configurations among known stationary obstacles. An efficient GA-PATH algorithm was designed to generate optimal global trajectories of the robot in 3-D space. The GA-PATH was required to optimize fitness functions formed from combinations of performance indices such as obstacle avoidance, total joint angle movement minimization and the robot arm configuration checking. The simulations were carried out for a Mitsubishi RM101 robot model. The GA-PATH algorithm has shown its power for generating feasible collision-free paths, and as a powerful global planner. Examples of efficient solutions to difficult path planning problems which arise in a cluttered workspace are presented.

ACKNOWLEDGEMENTS

My sincere appreciation goes to my supervisor Dr. R.W. Toogood for his guidance, supervision and valuable advice throughout every phase of this program.

Thanks to my wife Yinghong Wang for her understanding, encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Techniques for programming robots have achieved enormous progress during the past ten years. As one of the central problems in robotics, the motion planning problem has been extensively studied both from theoretical interests and from practical requirements [Brady, 1982; Schwartz and Sharir, 1988]. The objective of path planning is to plan a collision-free path for a robot through a workspace populated with obstacles. Typically, given two configurations of a robot (i.e. a starting point and a goal point), a search is made for a collision-free path between the two given configurations. Additional criteria such as global short path, safe path or fast path may also be prescribed. Recently, new algorithms have been introduced for robot path planning with respect to a variety of constraints such as time, velocity, and obstacle detection. These approaches, incorporating evolutionary programming, genetic algorithms, or artificial neural networks, have demonstrated great application potential in the robotics field.

Many robot applications are based on a motion trajectory composed of a sequence of spatial displacements of a robot arm. In order to obtain full spatial flexibility, a standard robot system usually has six degrees of freedom, which brings serious non-linearities and strong couplings in the system. Therefore, the search space for a feasible path is non-differentiable, non-linear, time-variant and parameter dependant [Kwok, et al., 1993]. Conventional techniques for solving the basic path planning problem do not have full generality and sometimes lock on local optima. These drawbacks have lead to a number of studies on improving the traditional techniques and finding new algorithms.

1.1 Current Research on Robot Motion Planning

Recent developments in motion planning have grown rapidly in response to increasing industrial demand for automatic manufacturing systems. Modern manufacturing systems use more and more robotic manipulators; more autonomous and intelligent robots are anticipated in the future. Future robots are expected to possess advanced capabilities of sensing, planning, learning, and control, enabling them to gather knowledge about their environment, and use this knowledge in planning and carrying out tasks.

Current research in robotics aims to identify the basic capabilities that an autonomous intelligent robot system will need and to advance understanding of the mathematical and algorithmic principles fundamental to these capabilities. Techniques for the automatic planning of robot motions have advanced substantially, and have shown themselves to have significant mathematical content; tools drawn from classical geometry, topology, and algebraic geometry have all been used. Despite the minor differences, these methods are based on a number of general approaches including visibility graphs, cell decomposition, and potential field [Latombe, 1991]. Some of these methods are discussed more fully in the next chapter.

1.2 Genetic Algorithms in Robotics

Robot systems are systems that result from the integration of a medley of subsystems and feedback loops where literally hundreds of variables may affect their operation. The central problem in robot process control and path planning is that the optimal values of the system's control and configuration parameters are not known, and there is no straightforward algorithm to discover them. Traditional optimization techniques, such as random search or gradient methods, depend greatly on a well defined, deterministic relationship between the parameters and the resulting performance. Even though these techniques have improved, they are unable to construct an overall picture of the search space and to optimize the performance of very complex systems.

In the manufacturing environment, the successful application of conventional procedures is either in situations where the system can be modelled with sufficient accuracy, or when the number of plausible parameter values is small enough to be tested exhaustively. For many systems that cannot be adequately modelled, or whose state space is too large, an alternative technique is required to efficiently search for the system's optimal parameter values.

Generally, any efficient optimization algorithm must use two techniques to find a global maximum: exploration to investigate new and unknown areas in the search space, and exploitation to make use of knowledge found at points previously visited to help find better ones. These two requirements are contradictory, and a good search algorithm must find a trade off between these two.

Genetic Algorithms (GAs) are improvement search algorithms loosely based on the processes of natural selection in biological system. They utilize a "survival of the fittest" concept among string structures that contain coded representations of the system parameters. They merely need a way to test many trial solutions to ascertain quickly how well any one meets established criteria, i.e., "fitness" of the solution. The beauty of the GAs is that once the criteria for evaluating fitness for each trial solution are in hand, a simple and general algorithm searches in a trial-and-error fashion keeping track of the most fit solutions, building on them, and eventually producing the "best" possible solution. Unlike many methods (eg. gradient methods), GAs use probabilistic transition rules (as opposed to deterministic rules, eg. steepest descent method) to guide their search towards regions of the search space with likely improvement. This is what makes GAs robust strategies.

Due to the robust nature of GAs, several projects on the implementation of the GAs in the robotic field have been carried out in the past few years. The GAs have been applied to problems such as robot trajectory generation, obstacle avoidance, inverse kinematics of redundant manipulators, and reactive control parameters optimization for autonomous robotic navigation. Genetic Algorithms applied to such problems appear to be very promising while traditional optimization methods cause difficulties. Further discussion of GAs in robotics is presented in the next chapter.

1.3 Objective

The objective of this thesis is to determine the feasibility of the use of Genetic Algorithms for finding a collision-free path for the Mitsubishi RM101 robot between arbitrary start and goal configurations among known stationary obstacles in 3-D space.

1.4 Organization Of The Thesis

The thesis is organized as follows. Chapter 2 gives an overview of traditional robot trajectory generation approaches and the applications of GAs in the robotic field. The basic concept and working principles of GAs are described. Chapter 3 describes the Mitsubishi RM101 robot environment, such as the robot work space definitions, the Denavit Hartenberg joint limits of the robot, obstacle definitions and the algorithm for 3-D collision detection. Chapter 4 provides details on the GA-PATH algorithm used for the Mitsubishi RM101 robot trajectory generation. The results obtained from the simulations on 1-D, 2-D and 3-D cases are also provided. Finally, Chapter 5 summarizes the great power of the GA-based path planner and its potential for real world applications.

2. LITERATURE REVIEW

2.1 Traditional Robot Trajectory Generation Approaches

Research in robot motion planning can be traced back to the late 1960's. Nevertheless, most of the efforts are more recent and have been conducted during the 1980's. Over the last few years, the theoretical and practical understanding of some of the issues has increased rapidly.

Motion planning involves such diverse aspects as computing collision-free paths among stationary and possibly moving obstacles, coordinating the motions of several robots, planning sliding and pushing motions to achieve precise relations among objects and dealing with models of physical properties such as mass, gravity and friction. Therefore, motion planning requires the robot to consider geometrical constraints, as well as physical and temporal constraints.

The path planning problem plays an important role in the manipulation and navigation of robots. There are two general approaches which have been employed to solve the path planning problem. One is the *configuration space* method [Lozano-Perez and Wesley, 1979]. Here the original problem of planning the motion of an object through a space of obstacles is transformed into an equivalent, but simpler, problem of planning the motion of a point through a space of enlarged configuration space obstacles (C-obstacle). Configuration space provides an effective framework for investigating a variety of robot motion planning problems, and planning safe motions for the links of an articulated manipulator [Takahashi and Shilling, 1989]. The configuration-space technique is very effective when applied to purely translational motion in a plane where the optimal path can be readily found. However, when the moving object is allowed to rotate, the resulting configuration-space obstacles become more complex. In general, this approach works well when the object is not allowed to rotate; otherwise, this approach will face a major computational problem.

The second general approach to solve the path planning problem is to search the free space directly without first transforming the problem to configuration space [Brooks, 1983]. Explicit representations of free space using overlapping generalized cones called *freeways* are used. This method isolates free space in the form of generalized cones and the robot will traverse along the axes of these generalized cones. Translations are performed along freeways with rotations performed at the intersections of freeways. One useful feature of this approach is that it typically generates paths that stay well away from the obstacles, although in some instances this leads to paths which are considerably longer than the shortest path. When the workspace is only sparsely or perhaps moderately populated with obstacles, this method is very fast and quite effective. The principal drawback to this technique occurs when the workspace is cluttered with closely spaced obstacles. In this case the method often fails to find a safe path even when one exists.

Among the many methods which have been applied for solving the basic motion planning problem, roadmap, cell decomposition, and potential field methods are the three general approaches which were categorized by Latombe [1991].

2.1.1 Road Map Method

The key issue in this approach is the construction of the roadmap. The roadmap approach to path planning consists of capturing the connectivity of the robot's free space. Once a roadmap has been constructed, it is used as a set of standardized paths. Path planning is thus reduced to connecting the initial and goal configurations to points in the roadmap and searching it for a path between these points. The constructed path, if any, is the concatenation of three subpaths: a subpath connecting the initial configuration to the roadmap, a subpath contained in the roadmap, and a subpath connecting the roadmap to the goal configuration [Latombe, 1991].

2.1.2 Cell Decomposition Method

Cell decomposition methods are perhaps the most extensively studied motion planning methods so far. They consist of decomposing the robot's free space into simple regions, called cells, such that a path between any two configurations in a cell can be easily generated. A non-directed graph called the connectivity graph representing the adjacency relation between the cells is then constructed and searched. Its nodes are the cells extracted from the free space and two nodes are connected by a link if and only if the two corresponding cells are adjacent. The outcome of the search is a sequence of cells called a channel. A continuous free path can be computed from this sequence [Latombe, 1991].

2.1.3 Potential Field Method

A straightforward approach to motion planning is to discretize the configuration space into a fine regular grid of configurations and to search this grid for a free path. Since the grid is in general enormous, this approach requires powerful heuristics to guide the search. Several types of heuristics have been proposed. A widely used heuristic is guiding the robot along the directions of "forces" generated by a function which is interpreted as an artificial potential field. Based on this approach, the robot represented as a point in configuration space is treated as a particle moving under the influence of an artificial potential produced by the goal configuration and the C-obstacles. Typically the goal configuration generates an "attractive potential" which pulls the robot toward the goal, and the C-obstacles produce a "repulsive potential" which pushes the robot away from them. In this approach, the search strategy is computed by using a potential field defined in a three- or two- dimensional workspace, and therefore, the computational cost for the motion-planning problem mainly depends on that for defining this artificial potential field. It is well known that this method can eventually lead the search to local minima of the potential field and provides no way of escaping these minima. The main reason for this drawback is considered to be that there is no clear relationship between the search strategy derived from the potential field in the workspace and the situation of the free space defined in the N-dimensional configuration space. When the configuration space obstacles are explicitly represented, the potential field without such local minima may be defined. In this case, the moving object might be able to avoid the trapped situations [Latombe, 1991].

The roadmap and cell decomposition methods reduce the problem of finding a continuous free path to that of searching a graph (e.g. visibility graph, Voronoi diagram, connectivity graph) by first analyzing the connectivity of the free space. They are often used in 2-D space path planning situations. Unlike these methods, potential field methods move from one configuration in the grid to another at each step, basing their choice on the computation of the potential gradient. It is typical for this gradient to depend only on the contents of the configuration space in the neighbourhood of the current configuration of the robot. Therefore, potential field methods are often called local methods, while the roadmap and cell decomposition methods are called global methods.

The global and local approaches are two main ways to deal with path planning problems. The global path planning assumes complete knowledge of obstacle locations in planning a path from an initial position to a target position. The global approaches are complete in the sense that if a path exists it will be found. Unfortunately, computing the complete configuration space is very time consuming, and the complexity of this task grows exponentially as the number of degrees of freedom increases. The local approaches need only partial knowledge of the configuration space, the decisions to move the robot are taken using local criteria and heuristics to choose the most promising directions. Consequently, the local methods are much faster, but, they are not complete, it may happen that a solution exists and is not found. As many optimization techniques, the local approaches are likely to entrapment in some local minimum, where a path to the goal has not been found and from which it is impossible to escape.

2.2 Genetic Algorithm and its applications

All natural species survive by adapting themselves to the environment. This natural adaptation is the underlying theme of Genetic Algorithms. GA's search utilizes a Darwinian survival-of-the-fittest strategy to eliminate unfit characteristics exhibited by members of a population of solutions. GAs use random information exchanges, with exploitation of knowledge contained in old solutions, to effect an exploration search mechanism with surprising power and speed. This is a technique that is both global and robust over a broad spectrum of problems. As stated by Goldberg [1989]: "Many traditional schemes work well in a narrow problem domain. Enumerative schemes and random walks work equally inefficiently across a broad spectrum. A robust method works well across a broad spectrum of problems".

2.2.1 Genetic Algorithm in General

The interest in GAs has been growing rapidly since the early 1970s. The primary monograph on this topic is John Holland's *Adaptation in Natural and Artificial Systems* [Holland, 1975]. Genetic Algorithms are intrinsically different from conventional optimization techniques. As robust stochastic searching algorithms, GAs have the power of global exploration and exploitation. This class of methods is based on the mechanics of natural selection and the natural genetics that combines the notion of survival of the fittest and parallel evaluation of many nodes in the search space. GAs search for a population of points, not a single point, so that they can arrive at the globally optimal point rapidly and meanwhile avoid locking at local optimum. They work with a coding of parameter sets, not the parameters themselves, so that they can avoid the analytical limitations of search spaces. GAs only require to evaluate the objective function to guide their search, not derivatives, so that they can utilize various kinds of objective functions even multiple, nonlinear and non-analytic. They exploit probabilistic transition rules, not deterministic ones, so that the solution efficiently converges to the neighbourhood of an optimal solution. The drawback of GAs is that it cannot absolutely guarantee the search converges to a feasible solution even it may exists.

Goldberg [1989] summarized the correspondence between natural and artificial terminologies, such as chromosome in a natural system refers to a string in GA, gene refers to feature and allele refers to feature value. These terminologies will be referenced in the following chapters.

The main data structure of the GA is the population, which is simply a collection of chromosomes (members of a population are referred to as individual strings). The initial population (i.e. parameter sets for each individual) is normally chosen at random. In each generation of the algorithm, a new population is produced by mating individuals from the previous one. This reproduction of individuals is performed by a set of genetic operators. The method has proven to be able to search very high dimension spaces. The fitness function is treated purely as a "black box", the only information required being its value at different points in the search space.

Generally, a GA uses three fundamental genetic operators: reproduction, crossover and mutation. Given an optimization problem, a GA encodes the parameters concerned into finite length strings, each of which presents a possible solution to the problem, and then works with a set of strings, called the population. The GA combines the three operators and a fitness function evaluation mechanism to guide the search. The GA performs the basic tasks of copying strings, exchanging portions of strings as well as changing some bits of strings, and finally finds and decodes the solution to the problem from the last pool of mature strings. The working principles of a simple GA can be explained as follows:

(1) **Representation:** The most common representation technique is binary representation which has been shown to be the optimal one [Holland, 1975]. The basic component treated by a GA is the binary string formed by concatenating substrings which are encoded from the parameters of a problem. The length of each substring can be different, depending on the optimal objective of a problem.

(2) **Initialization:** The starting population is created by randomly generated bit strings. The population size is an important factor that effects both ultimate performance and the efficiency of a GA. A large population discourages premature convergence to sub optimal solutions but decreases the rate of convergence.

(3) **Evaluation:** The evaluation is the link between the genetic algorithm and the problem to be solved. A fitness function must devised for each individual problem. When the current population is to be evaluated, each string of the current population is decoded into the corresponding parameters. Then, the fitness function takes these parameters as input and returns a fitness value that is a measure of this parameter set's performance on the problem to be solved. Along with the coding scheme used, the fitness function is the most crucial aspect of any GA [Beasley, Bull and Martin, 1993].

(4) Reproduction: This is a process in which the strings are copied according to their fitness values. The strings with higher fitness values have higher probabilities to produce one or more copies in the next generation. In a policy called elitist reproduction, the current best string is guaranteed a long life from generation to generation. These reproduced or copied strings are placed in a mating pool where the action of crossover and mutation may take place.

Parent selection is the task of allocating reproductive opportunities to each individual. In principle, individuals from the population are copied to a "mating pool", with highly fit individuals being more likely to receive more than one copy, and unfit individuals being more likely to receive no copies. Under a standard generational replacement scheme, the size of the mating pool is equal to the size of the population.

The behaviour of the GA very much depends on how individuals are chosen to go into the mating pool. There are several different methods for determining how effectively a given string "competes" in the reproduction process. The reproduction operator may be implemented in algorithmic form in a number of ways. Perhaps the easiest is to create a biased roulette wheel where each current string in the population has a roulette wheel slot sized in proportion to its fitness. For example, a string i gets selected with probability $p_{(select)i} = f_i/\sum f_i$. Where i is a string index, f_i is the string's fitness and $\sum f_i$ is the total fitness of the population. However, as pointed out by De Jong [1975], such a selection process will cause a stochastic error, i.e., there is a chance that a good string may not be selected for mating. Roulette wheel selection is a high-variance process with a fair amount of scatter between expected and actual numbers of copies.

A number of selection alternatives have been investigated in an effort to reduce the stochastic errors associated with roulette wheel selection. Deterministic sampling is a scheme where the probabilities of selection are calculated as usual, $p_{(select)i} = f_i / \Sigma f_i$. Then the expected number of individuals for each string i is calculated $n_i = p_{(select)i} \cdot n$ (n is population size). Each string is allocated samples according to the integer part of the n_i values, and the population is sorted according to the fractional parts of the n_i values. The remainder of the strings needed to fill the population are drawn from the top of the sorted list [Goldberg, 1989].

The stochastic remainder selection sampling methods start in a manner identical to deterministic sampling. The integer part of the product is the number of copies of the individual being reproduced. In stochastic remainder sampling with replacement, the fractional parts of the expected number values are used to calculate weights in a roulette wheel selection procedure that is then used to fill the remaining population slots. In stochastic remainder sampling without replacement, the fractional part of the product is treated as a probability of having another copy of the individual being reproduced [Goldberg, 1989].

In stochastic tournament procedure, selection probabilities are calculated normally and successive pairs of individuals are drawn using roulette wheel selection. After drawing a pair, the string with higher fitness is declared the winner, inserted in the new population, and another pair is drawn. This process continues until the population is full [Goldberg, 1989]. These strategies, in which good strings get more copies in the next generation, all emphasize the survival-of-the-fittest concept of GAs.

(5) Crossover: Crossover is a process in which two individuals are arranged to swap their chromosome strings at some randomly chosen position(s), creating two new strings, if the probability test is passed. This process can produce new strings that are radically different from their parent strings. The standard GA uses a single crossover point. It is implemented by mating two members of the newly reproduced strings at random. A crossover position K in the strings is selected at random between 1 and the string length L minus 1 [1, L-1]. Two new strings are then created by swapping characters between position 1 through K and position K+1 through L. For example, consider string 1 and string 2 from a population i, and suppose the cross-point is randomly chosen to be 5. The results of the crossover operation are two new strings as shown below:

String 1: 11111 11	New String 1: 11111 00
String 2: 00000 00	New String 2: 00000 11

Crossover is a remarkably important operator of a GA. It is regarded as the distinguishing feature of GAs from all other optimization algorithms and as a critical accelerator of the search process when a GA runs. Although the crossover operation is a randomized event, when combined with reproduction it becomes an effective means of exchanging information and combining portions of good quality solutions. Reproduction and crossover give GAs most of their search power.

Traditionally, genetic algorithms have relied upon 1 and 2-point crossover operators. However, many different crossover algorithms have been devised involving more than one cut point. An advantage of having more crossover points is that the problem space may be searched more thoroughly. Researchers now agree that 2-point crossover is generally better than 1-point crossover [Beasly, D., D.R. Bull and R.R. Martin, 1993].

Many recent empirical studies, however, have shown the benefits of higher numbers of crossover points. Some of the most intriguing recent work has focused on uniform crossover, which involves on the average L/2 crossover points for strings of length L. Spears and De Jong [1991] presented a framework for understanding the virtues of parameterized uniform crossover. The standard form of uniform crossover swaps two parents' alleles with a probability of 0.5. They found two important virtues of uniform crossover. The first was the ease with which the disruptive effect of uniform crossover can be controlled by varying the probability of swapping. This is useful in achieving the proper balance between exploration and exploitation. The second virtue was that the disruptive potential of uniform crossover does not depend on a string's defining length. This allows uniform crossover to perform equally well, regardless of the distribution of important alleles. Furthermore, the standard uniform crossover has a higher recombination potential than other crossover operators (Recombination potential is defined as the ability of crossover to create higher order hyperplanes when the parents contain the necessary lower order hyperplanes)[Syswerda, 1989].

(6) Mutation: This is a process in which a single bit in a string is "flipped" to the opposite value if a probability test is passed. The mutation operator is largely a background noise to protest against the focusing effect of the recombination operator. The mutation process may quickly generate new strings which might not be conveniently produced by the previous reproduction and crossover processes. The mutation operator changes the characters in an individual string on a bit by bit basis. Although mutation is necessary, sometimes it may suddenly spoil the opportunity of the current appropriate generation. Therefore, the mutation probability compared with the crossover probability is quite low (Usually around 1/n). Mutation brings in new possibilities for improvements and takes care of some of the lost information during crossover and reproduction.

After mutation the fitness values of the new population's strings are evaluated, and the cycle of reproduction, crossover and mutation repeats. This process continues until a predefined convergence criteria is met by the "best" string or by the population average fitness.

(7) Iteration: The GA runs iteratively repeating steps (3) to (6) until the search reaches the ending conditions. Figure 1 shows a standard GA processing flow chart.

Genetic Algorithms have been successfully applied to solve many complicated problems [Davis, 1991], such as function optimization, design of a communication network, the VLSI circuit layout, image classification and pattern recognition, robot trajectory generation, machine learning, schedule optimization, travelling salesman problem and so on.

Defining the type of problem that is well fitted for the GAs is one of the most important questions related to the applicability of GAs. Davis [1991] provided an overview on the problem domain that is suitable for the application of genetic algorithms:

the domain should contain multiple extrema; the domain should exhibit some epistasis (an epistasis is the inhibition of one part of a solution by the action of another).

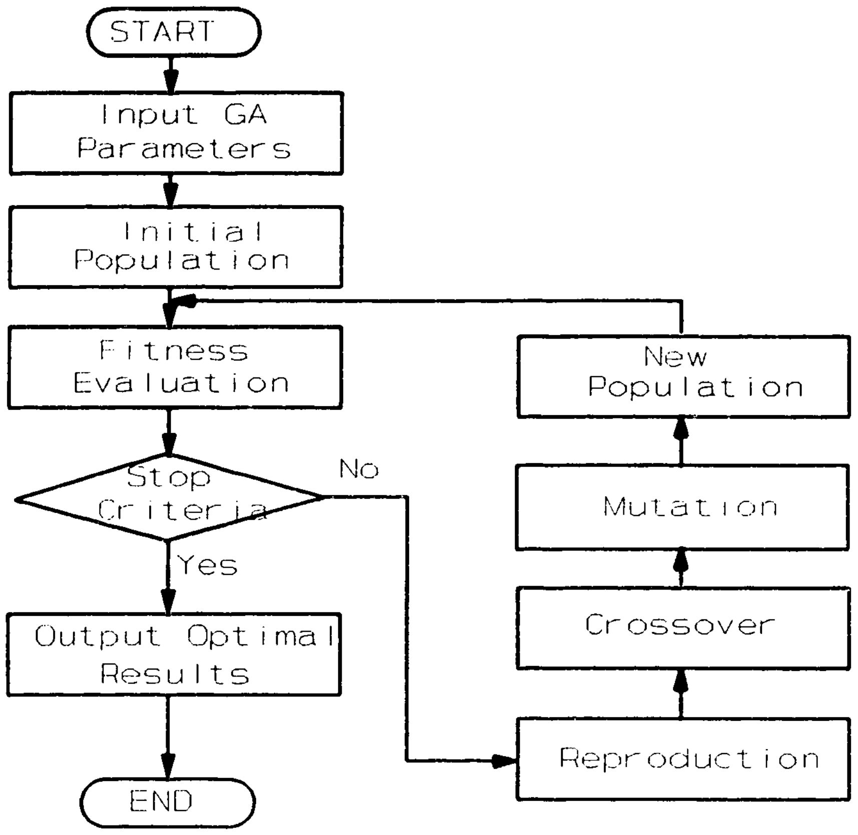


Figure 1. A Simple Genetic Algorithm Flow Chart

GAs have been theoretically and empirically proven to be able to provide robust search in complex spaces. The manipulator path planning problem is a nonlinear optimization problem with nonlinear, nonconvex constraints and unconnected feasible regions. The huge search space, non-linearity, non-analytic and very complex objective function characteristics of the manipulator path-planning problem seems to fit the applicability of Genetic Algorithms. Therefore, applying GAs to solve the path planning problem is very promising.

2.2.2 Applications of Genetic Algorithms in Robotics

A robotic system can provide a flexible mechanical structure which can perform a wide variety of tasks in a reprogrammable fashion. The system is generally characterized as a highly complex, nonlinear, tightly coupled, and multi-dimensional system. GAs applied to optimize such a system may bring great benefit for the system's performance. One area in which GAs can improve the robot performance is in optimizing trajectory specification. Trajectory specification has been left to the human operator in most robot applications, and the human interface presents a major problematic aspect of robot applications [Davidor, 1991].

Redundant manipulators have been recommended by many researchers to increase flexibility and dexterity around restricted workspace in the presence of obstacles. They further provide a better ability to avoid singular configurations and the excessively high joint velocities and accelerations encountered at singularities.

In the following, six applications of the GAs in robot trajectory planning will be reviewed. The applications use quite different methods for problem definition and parameter encoding, and fitness evaluation. Comparison of these methods show the versatility of GAs in solving the path planning problem.

Davidor [1991] applied a Trajectory-GA algorithm to robot trajectory generation in 2-D space. A simulated redundant robot structure was used in his experiment as shown in Figure 2. The system is a 3-link planar robot. An arm-configuration $\Theta=(\Theta_1,\Theta_2,\Theta_3)$, is defined by the three link positions incorporated in the structure. Given the triplet of link positions and length of each link, the end-effector's position is uniquely determined.

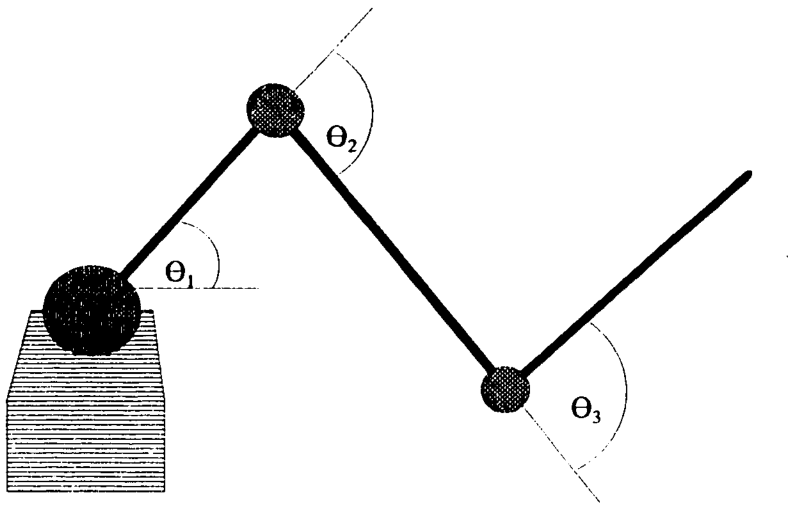


Figure 2. A 3-link Planar Robot Configuration

The trajectories are formed by joining several arm-configurations,

$$\{(\Theta_1,\Theta_2,\Theta_3)_1(\Theta_1,\Theta_2,\Theta_3)_2 \ldots (\Theta_1,\Theta_2,\Theta_3)_i \ldots (\Theta_1,\Theta_2,\Theta_3)_n\}$$

where $i=1,2\ldots n$ designates the order of execution according to the ascending value.

Based on Davidor's algorithm, a trajectory could be represented by strings of varying lengths to allow for a varying number of triplets to describe a trajectory, and the representation contains repetitive sub-structures which must not be separated.

A preselection mechanism [Cavicchio, 1970] was adopted for the selection mechanism . In this case, because there are only one trajectory and two parents, so the best two among the three were chosen to remain in the population. An analogous crossover was introduced due to the order dependency character of the representation.

The evaluation mechanism is based on the absolute performance of the trajectory and the discrepancy between the starting and terminating ends of the desired path and the corresponding ends in the actual path, i.e.,

$$D_t = |(E_{desired})|_{begin} + |(E_i - E_{desired})|_{end} + \int_{trajectory} |deviation|$$

Where E_i is actual end-effector position, $E_{desired}$ is desired end-effector position. An exponential scaling was used so that,

$$\text{fitness} = 100 * \exp^{-D_t}$$

Because a problem is usually not precisely articulated, nonlinear scaling potentially contains a risk of introducing bias into a GA [Crow and Kimura, 1970]. The justification for using such a powerful scaling function is founded on the argument that a practical method with meaningful and fast improvements in performance is preferred [Davidor, 1991].

The results presented by Davidor demonstrated the power and robustness of the trajectory-GA, and the analogous crossover proved to be capable of handling order dependant representations of varying length. The efficiency of the trajectory-GA, compared to that of random search and hill-climbing algorithms was also investigated and it was found that not only the trajectory-GA works, but it works well. The trajectory-GA optimized trajectories efficiently, and more importantly, reliably. Besides these conclusions, no further statistical comparison was given by the author.

Khoogar and Parker [1991] employed GAs for a robot path planning problem in a 2-D space. They used a planar 3 DOF robot. The extra degree of freedom here was used to plan collision free motions of the robot from an arbitrary starting point to a desired goal point. In Khoogar's work, the encoding method was quite interesting as described below.

The configuration transformation from the starting point to the goal point is obtained through N incremental moves which are assumed to take a relatively small finite value. The number of moves N is selected by the algorithm according to the requirement to accomplish the task. Incremental joint moves are coded with ternary numbers (-1, 0, 1), where -1 represents a small rotation (degrees) in the negative direction, 0 represents no move, and 1 represents a small rotation (degrees) in the positive direction. Therefore for an 3 DOF robot a set of joint incremental moves can be represented with a 3 bit ternary number, and a 3*N ternary number can represent N successive moves all coded within a single string,

$$\{(\Theta_1,\Theta_2,\Theta_3)_1(\Theta_1,\Theta_2,\Theta_3)_2 \ldots (\Theta_1,\Theta_2,\Theta_3)_i \ldots (\Theta_1,\Theta_2,\Theta_3)_N\}.$$

Where $i = 1,2,\ldots N$ designate the number of moves.

The genetic algorithm was used to find a 3*N ternary string such that the combination of the N moves will guide the end-effector towards the desired final point through a collision free path, if the path exists.

The performance criteria were designed according to (1) the motion should guide the end-effector to the goal point, and (2) the motion must be collision free. The fitness function was therefore designed to reflect the above criteria as follows:

(1) Minimized the position error: the end-effector's position after N moves was calculated using the forward kinematic equations, and the distance between the end-effector position and the desired goal point was calculated as:

$$\text{Error}_1 = \{(GPx\text{-}EPx)^2 + (GPy\text{-}EPy)^2\}^{1/2}$$

where GPx and GPy are the X and Y coordinates of the goal point, and EPx and EPy are the X and Y coordinates of the end-effector position after the N moves. Note that the algorithm does not ensure the robot reaches the goal point after N moves. In such a case, the GA was restarted with the new start point set to the end point of the previous run.

(2) Obstacle avoidance: Khoogar and Parker used the Turbo Pascal graphics package to check for collisions between the obstacles and the robot linkage by simulating the individual link movements. The obstacles and the robot were drawn on the graphics screen in different colours. The number of graphics pixels required for each obstacle was then calculated. After all N robot moves were drawn, the number of original colour pixels in the obstacles was determined. If this number does not match the original

values for a given obstacle, then a collision has occurred. If there is a collision then a relatively large value is added to the fitness value, in order to penalize the string. Otherwise if there is no collision the error remains unchanged.

$$\text{Error}_2 = \text{constant} \quad \text{if there is a collision}$$

$$\text{Error}_2 = 0 \qquad \text{otherwise}$$

(3) Total Error was then given by the sum of the two components:

$$\text{Error}_{\text{total}} = \text{Error}_1 + \text{Error}_2$$

Tournament selection method, standard one point crossover and mutation operators were used. It was shown that when the robot arm was situated such that improvements in reaching the goal point were practically impossible, additional heuristic decision making routines were employed in order to prevent the robot from getting trapped through un-realizable paths. In these situations intermediate goal points were introduced, and the robot was forced to pull back from its initial heading and to take an alternative path. The simulation results show that the developed GA was a very simple and useful method which only used the forward kinematic equations to find the incremental motion of the joint variables in order to obtain a collision free inverse kinematic solution [Khoogar and Parker, 1991]. Nonetheless, this method relied on some ad hoc decisions which would rescue the GA if a complete trajectory could not be found.

Parker and Goldberg [1989] also applied GAs to an inverse kinematics problem in which a redundant robot's maximum joint displacement in a point-to-point positioning task was minimized. The robot had four degrees of freedom, which allows for an infinite number of joint solutions for arbitrary positioning of the end-effector within the three-dimensional workspace. The basic dimensions of the robot were taken from the PUMA 566 industrial robot arm [Unimation 1987].

The redundant robot end-effector was assumed to be at some initial position, $P_i=(X_i\ Y_i\ Z_i)^T$, with known initial joint angles $\Theta_{1,i}$, $\Theta_{2,i}$, $\Theta_{3,i}$, $\Theta_{4,i}$. The desired final position of the end-effector was specified as $P_f=(X_f\ Y_f\ Z_f)^T$. The fitness function selected for GA combined two terms: (1) arm positioning error; and (2) joint angle displacements from the initial position. The positioning error was calculated as

$$\text{Error}_p=\{(X_f-X_i)^2+(Y_f-Y_i)^2+(Z_f-Z_i)^2\}^{1/2}$$

The additional rotation displacement was determined by,

$$\text{Displacement}_{Max}=\text{Max}\{(\Theta_{1,f}-\Theta_{1,i}),\ (\Theta_{2,f}-\Theta_{2,i}),\ (\Theta_{3,f}-\Theta_{3,i}),\ (\Theta_{4,f}-O_{4,i})\}$$

The maximum individual joint rotation from the initial position as a "cost" was selected, and the two error terms were combined into a single fitness function.

$$\text{Fitness} = \text{Error}_p + K*\text{Displacement}_{Max}$$

where the factor K scales the contributions from each term. In this application the GA minimized the fitness function by forcing the positioning error and the maximum rotational displacement to a minimum value.

A 40-bit string was used in the GA where each of the four joints was assigned to 10 bits. The joint angles could then be decoded from the values of their 10 bit string by the following (PUMA 566 industrial robot).

Θ_1 = -160° + (bit value/1024)*320°

Θ_2 = -225° + (bit value/1024)*270°

Θ_3 = -45° + (bit value/1024)*270°

Θ_4 = -45° + (bit value/1024)*270°

The standard GA was applied to compute the joint angles that would position a robot at a target location while minimizing the largest joint displacement from the initial position. They concluded that although the final positioning accuracy was not ideal, GAs were suitable for off-line programming of a redundant robot in point-to-point positioning tasks.

Ram, Arkin et al. [1994] applied GAs to the learning of local robot navigation behaviours for reactive control system. They employed GAs to optimize robot navigation control parameters in the system. The GAs were used as an unsupervised learning method for a reactive control architecture which greatly reduced the effort required to configure a navigation system. Unlike standard genetic algorithms, a floating point (rather than binary) method was used for the parameter representation.

Three schemata (move-to-goal, avoid-obstacle, and noise) were considered for their experiments. The schema parameters controlling the behaviour of these schemas were determined autonomously using a Genetic Algorithm. Each robot was given a fixed set of parameters which control its behaviour. The robot was then run through the simulated environment and its performance evaluated. New parameters were then generated and given to another generation. A GA-ROBOT algorithm was developed to learn optimum schema parameters under different conditions. The method was applied to a robot simulation in a two-dimensional world with stationary obstacles and a single goal position.

The task of each robot in a simulation was to move from the start to the goal point, while avoiding obstacles along the way. Robots were rewarded based on their traversal time, distance travelled, and number of collisions.

The algorithm started by generating a population of robots, an environment containing obstacles, and a single goal. Each of the robots was initialized with a set of randomly generated values for the motor schema parameters. In each generation, the robots were run through the environment three times. The robots moved through the environment until they either reached the goal or exceeded the maximum number of allowed steps. Optimal robots maximized safety, speed, or efficiency. Each of these was multiplied by a weighting factor and summed to give the raw fitness,

$$\begin{aligned} \text{Raw.fitness} = {} & \text{Collision.weight} * \text{Number.of.collisions} \\ & + \text{Time.weight} * \text{number.of.steps} \\ & + \text{Distance.weight} * \text{Distance.travelled} \end{aligned}$$

The raw fitness function provided weights for collision, time, and distance. These factors were combined into three robot types: safe, fast, and direct. A safe robot was optimized to avoid hitting obstacles. While both avoid collisions, fast robots prioritized speed whereas direct robot preferred shortest trips.

The crossover and mutation operators were modified to work with the floating point representation.

Ram and Arkin concluded that GAs provided a powerful method for searching for near-optimal solutions in complex search spaces, and it could be applied to the learning of robotic coordination in reactive control systems. Genetic Algorithms allowed not only rapid discovery of good behavioral parameters, but also tuning of parameters to varying robot and environmental conditions. An attractive feature of this approach was that the designer need only specify the robot and the environment; a deep understanding of the robot's behaviour and environmental interactions was not required. The algorithm would abstract the salient features of the environment as it searches for optimal parameters. The "fast" robots in their simulations learned to find wide paths which enabled greater speeds, the "safe" robots learned to avoid obstacles, and the "direct" robot learned to find the shortest path.

Shibata and Fukuda [1993] proposed an approach for multi-agent system coordinative motion planning by using GAs and fuzzy logic. In the proposed approach, each robot planned its motion while considering the known environment and using empirical knowledge for the unknown environment which includes the other robots. The GA was applied to optimize the planning of the motion. Through iterations, each robot acquires knowledge of its unknown environment expressed by fuzzy logic.

A path for a mobile robot was encoded based on the order of via points. Each robot had a starting point and a target point in the graph under the assumption that each robot passes each point only once or not at all. Each node in the graph had a number and the nodes were used to encode a path as a string which was expressed by the order of the numbers. These points were selected randomly at first, while adjacent numbers must be connected with a link in the graph. Since order based strings were used, specialized operations of crossover and mutation were implemented.

In the crossover, a number in the first parent was randomly selected at first. If the second parent had the same number as that in the first parent, each string exchanges the part of strings after that number. If not, another string was selected as the second parent, and the same operation was repeated.

After the crossover, the children were checked whether each string had same numbers. If so, a part of the string between the same numbers was cut off.

In the mutation, a position in each string was selected at random based on the probability of mutation. Then, the number of nodes was selected randomly for following positions which were connected sequentially.

To calculate the fitness, the length of the link between numbers in the string and the estimated cost at the public resources were calculated as:

$$\text{Fitness} = D_i + C_i$$

where D_i was equal to length of the link in the static map, and C_i was the estimated cost at *ith* public resource expressed in a dynamic map. Since C_i depended on time, the search space was huge and the conventional graph search techniques could not be simply applied to this search problem. However, the proposed approach can easily be applied while considering the cost C_i in the scalar fitness function. Since the GA searches multiple points in the search space at the same time, the GA could obtain the optimal and feasible solutions.

Zhao, Ansari et al. [1994] addressed a path-planning problem for a mobile manipulator system by using GAs. The problem was to find an optimal sequence of base position and manipulator configurations for performing a sequence of tasks. Task specifications were characterized by the required end-effector locations in a 3-D space and minimum force capabilities in specific directions. The formulation of the problem was nonlinear. The feasible regions for the problem were nonconvex and unconnected. The position of the manipulator end-effector was described by:

$$X_{tip} = X_{base} + X_{man}(\Phi)$$

where X_{tip} was a three-dimensional vector specifying the location of a task with respect to a global cartesian reference frame, X_{base} was the location of the mobile base with respect to the global frame, and $X_{man}(\Phi)$ was a vector function that maps the manipulator configuration vector Φ into the end-effector position relative to a cartesian reference frame fixed on the mobile base. A cost function was introduced that measures the energy or time required to move the system through the complete sequence of tasks. In general the problem was defined as follows:

$$\min\{C\} = \min\{\alpha C_{base} + \beta C_{man}\}$$

where C_{base} was the cost of moving the base through the complete sequence of n tasks. Likewise, C_{man} was the cost of having the manipulator taken into various configurations in completing the n tasks. α and β were weight factors for the cost of moving the base and the cost of changing the configuration of the manipulator, respectively.

The manipulator configuration Φ (Θ_i's) was coded and concatenated into a binary string. It was claimed that by encoding the manipulator configuration parameters into binary strings and by incorporating a moderate violation penalty into the fitness function, the search result was guaranteed to produce feasible solutions.

Their simulation system was a 3 DOF arm mounted on a 2 DOF mobile base. The joints with angles Θ_1, Θ_2, Θ_3 were refereed to as the base, shoulder, and elbow, respectively. Using a genetic string of 80 bits to encode the four parameters for performing two tasks with the locations and minimum oriented force capabilities, two parameters (Θ_1 and Θ_2) for each task, each parameter was thus represented by 20 bits (having encoded Θ_1 and Θ_2 for each task, Θ_3 can be obtained by using forward kinematic equations of the system). The feasible regions in which the mobile base can be located to perform the tasks were determined exhaustively.

A tangent function was employed for fitness scaling to accentuate the relative significance among population members to reward the good strings more as the population was converging. Stochastic remainder sampling without replacement, standard crossover and mutation were used for their simulation.

Simulation results demonstrated that the performance of the genetic algorithm greatly depends on the proper formulation of the fitness function. By comparing the performance with an exhaustive search algorithm, they concluded that the CPU time of the proposed genetic algorithm was roughly two orders of magnitude faster and the GA was an effective and efficient way to obtain a near optimal solution to the mobile manipulator path-planning problem.

Based on the many successful applications of Genetic Algorithms in robotics, this thesis will address the issue of 3 DOF robot path planning via Genetic Algorithms in configuration space. The reasons for introducing GAs to the 3 DOF manipulator path planning are (1) a 3 DOF manipulator can provide basic positioning capability, and the studies on the manipulator have more generality; (2) due to the search power and global optimization nature of GAs, there is a belief that GAs might prove to be a rewarding strategy to employ in solving path planning problems which existing techniques have a hard time to solve. Even where existing techniques work well, improvements may be made by hybridising them with a GA.

Many robots use a joint interpolated control scheme which produces coordinated simultaneous movement of several robot joints rather than a sequential or uncoordinated one. This coordinated motion results in the robot path to follow a straight line in joint space when directed to move to another pose. It is therefore much easier to calculate a path in the joint coordinate system (ie.a straight line). The reduction of the complex and time varying shape of a robot to a point is the motivation of adopting the configuration space approach for this research.

3. THE MITSUBISHI RM101 ROBOT ENVIRONMENT

This chapter will present the Mitsubishi RM101 robot working environment. The robot working space and obstacle definitions are defined here. The representation of obstacles in configuration space, the collision detection algorithm, and path visualization are also described briefly.

3.1 Mitsubishi RM101 Robot Work Space Definitions

The Mitsubishi RM 101 robot is a five link robot arm (Figure 3). The first three degrees of freedom are defined by three joint angles Θ_1, Θ_2 and Θ_3 which are specified by the Denavit-Hartenberg convention for robot kinematics. The wrist degrees of freedom (Pitch and roll) are ignored here. The first link is connected to the ground at the origin of the coordinate system, with Θ_1 defined as the rotation of this link around the Z axis. The angle Θ_2 defines the angle between the second link and the X-Y plane while Θ_3 defines the angle between the third link relative to the second link. Note that the second and third link always move in the same plane for any particular Θ_1.

The length of the shoulder, upper arm, and lower arm are predefined constants of 214 mm, 200 mm, and 250 mm respectively. The lower arm dimension includes the gripper. The limiting Denavit-Hartenberg angles for the Mitsubishi RM101 are shown in Table 1.

Table 1. Denavit-Hartenberg joint limit for the Mitsubishi RM101 Robot

Robot Joint	Positive Angle (Degree)	Negative Angle (Degree)
θ_1	120	-120
θ_2	120	-30
θ_3	0	-120

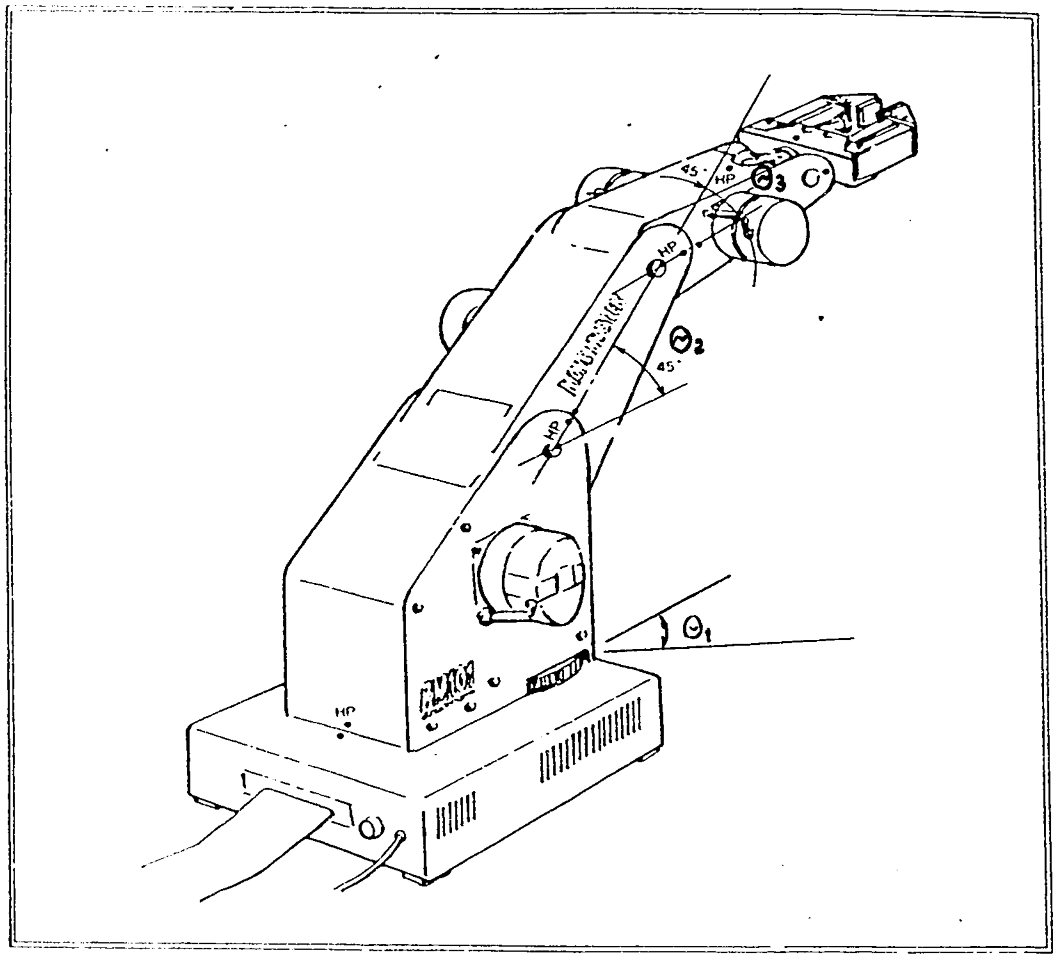
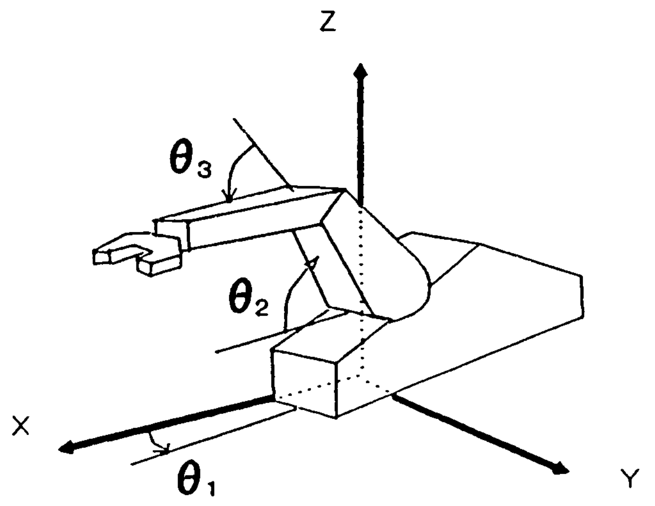


Figure 3. The Mitsubishi RM101 Robot

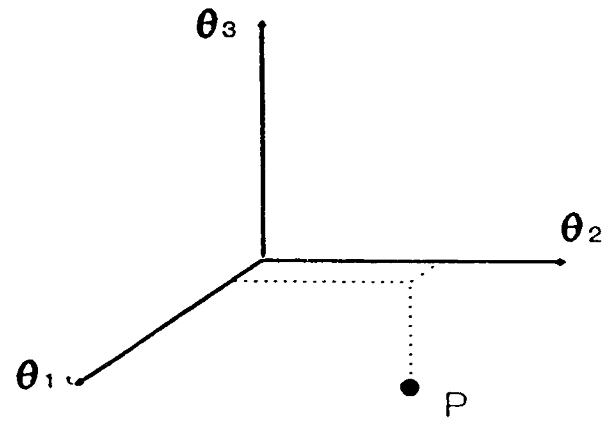
Considering only the first three joints, the robot's configuration is determined by the three joint angles, either given directly or found by using the inverse kinematic solution for a given set of X, Y and Z coordinates of the gripper. This configuration is represented by a single point in joint space, as shown in Figure 4. The position of the RM101 shown in Figure 3($\Theta_1=0$, $\Theta_2=45$, $\Theta_3=-45$) is known as the HOME position. The internal robot controller references all joint angles from this position. However, the path planning program GA-PATH uses the Denavit-Hartenberg angles which defines the horizontal as being the zero degree po on for the second link (i.e. Θ_2).

3.2 Obstacle Definitions

The origin for both the cartesian and joint coordinate systems is located at (0,0,0). Obstacles are defined as 6-sided rectangular polygons. The edges and planes on these obstacles always run parallel to the world axes. Because any obstacle can be represented by rectangular dimensions, rectangular parallelepiped obstacles were chosen for the sake of simplicity. If desired, a better approximation to an irregularly shaped obstacle can be achieved by the use of a number of adjacent rectangular boxes. The numbering and orientation of the vertices for an individual obstacle are shown in Figure 5. The lower left vertex of the obstacle is used as the reference. From this reference vertex, the length of the obstacle in the positive X, Y and Z direction must then be entered. The reference point will always be vertex 1 as in Figure 5. This is because the program calculates the location of the other vertices using the positive length entered for each axis direction.



(a)



(b)

Figure 4. A Robot Pose a) in World Space and b) in Joint Space

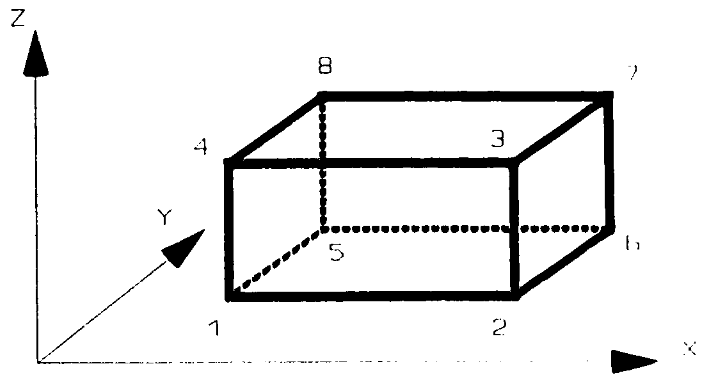


Figure 5. Obstacle Vertex Numbering Scheme

Shrinking the robot links to line segments and increasing the size of the obstacles, which takes into account some finite dimensions of the robot arm, is a fairly common procedure in obstacle avoidance research. This technique considerably reduces the complexity of the mathematics describing the geometry. The GA-PATH program utilized this concept and treats each robot link as a single line segment running along the centre-line of the link.

3.3 Obstacles in Configuration Space

In order to visualize the shape of the obstacle(s) in configuration space, the collision points of the robot with obstacles in the work cell are mapped into a joint space.

Note that in some robots, the goal point may not be unique due to kinematic redundancy. However, since Θ_3 in the Mitsubishi RM101 can only be negative, a unique kinematic solution does exist.

MAKEDATA [Wong, 1992] is used as a pre-processor for the program GA-PATH. The basic algorithm utilized in MAKEDATA is an exhaustive search collision detector. For any given set of joint angles (Θ_1, Θ_2 and Θ_3), a collision detection subroutine returns a flag signalling whether a collision with an obstacle or ground exists. By scanning at intervals throughout the available joint space, a complete representation of the obstacle in joint space was generated. Figures 6 and 7 show the views of two work cells containing (2 and 5) obstacles. With each figure is shown the appearance of the obstacle in joint space.

The required inputs to MAKEDATA are the reference vertex coordinates in the cartesian coordinate system, the length, width and height of the obstacle in the positive X, Y, Z directions respectively. The generation of the obstacles in configuration space on the computer screen is achieved by layering a series of contours on top of each other to create the 3-D effect. Each contour is for one constant Θ_3 plane, and the points that map out each contour are the boundary collision points, Θ_1 and Θ_2 for that particular Θ_3 plane. By stacking together a number of these boundary slices in the Θ_3 direction, a rough 3-D projection of the obstacle in joint space is achieved. In the program MAKEDATA, only the collision points that corresponds to a boundary collision point need to be stored. It should be stated, that this data is not used by the path planner, which detects collisions independently of the MAKEDATA file. Rather, this data is used as a visual aide in understanding the joint space obstacle configuration and observing the performance of the path planner.

3.4 Collision Detection

When the obstacle data is entered, the GA-PATH proceeds to define each of its six sides as an infinite plane in cartesian space using three of the vertices on that particular plane within the subroutine PLANE. The equation of a plane is defined as [Parkin, 1991],

$$AX + BY + CZ + D = 0 \tag{1}$$

where A, B, C, D = constant coefficients. Each particular set of joint angles Θ_1, Θ_2 and Θ_3, defines a unique configuration of the robot links which are treated as infinite lines in three dimensional cartesian space within the subroutine LINES. Such a line will have two equations to define it completely. In symmetric form, the equations are of the form

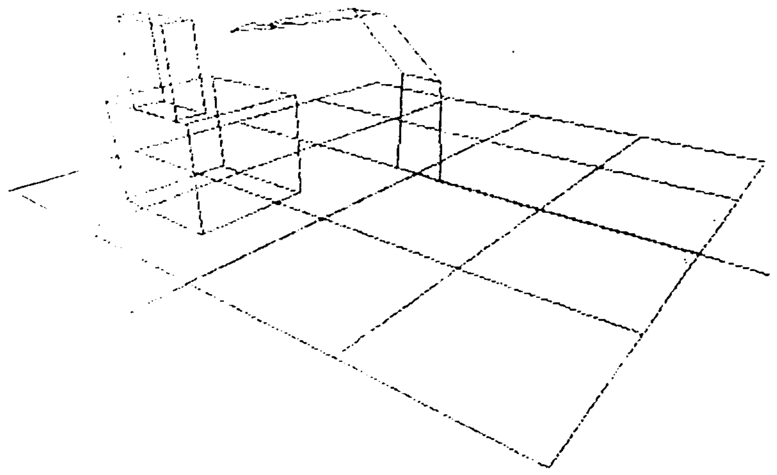
$$(X-X_1)/a = (Y-Y_1)/b = (Z-Z_1)/c$$

where a, b and c are direction numbers (non-zero) equal to (X_2-X_1), (Y_2-Y_1) and (Z_2-Z_1) respectively. The start and end coordinates for each link define the X_1, Y_1, Z_1 and X_2, Y_2, Z_2 values respectively. The above equations can be rearranged as

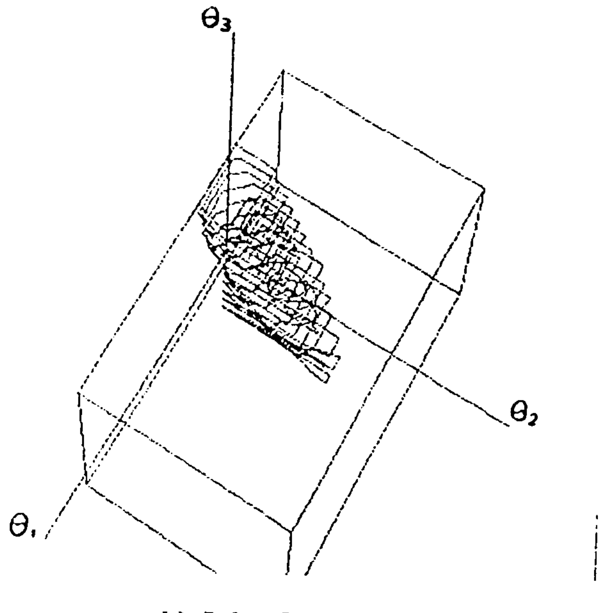
$$a_1X + b_1Y + c_1Z + d_1 = 0 \tag{2}$$

$$a_2X + b_2Y + c_2Z + d_2 = 0 \tag{3}$$

where a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2 are constant coefficients. Thus, by solving the above

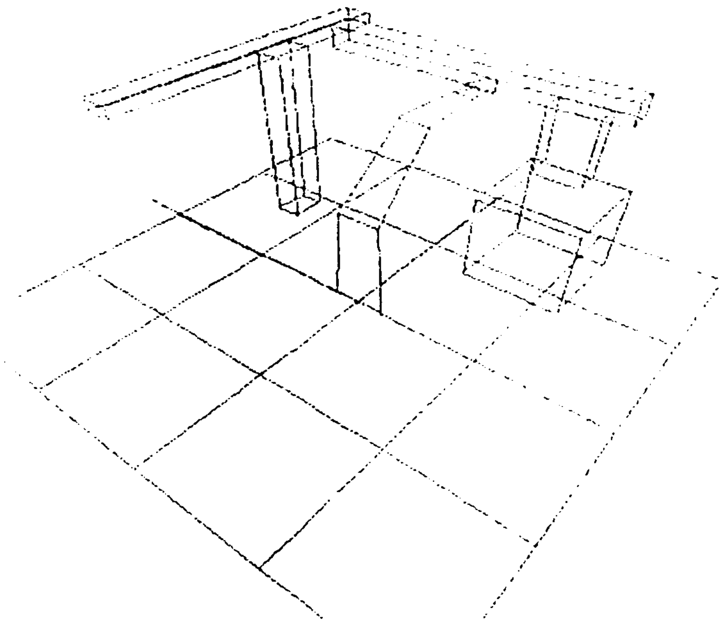


a) World Space

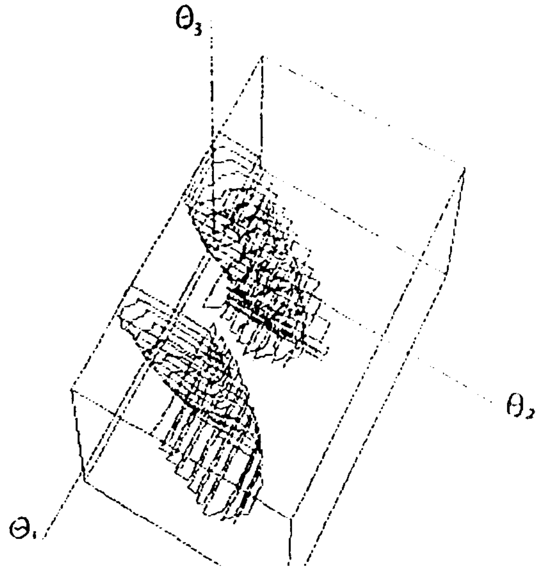


b) Joint Space

Figure 6. View of 2 Obstacles in a) World Space and b) Joint Space



a) World Space



b) Joint Space

Figure 7. View of 5 Obstaclǝs in a) World Space and b) Joint Space

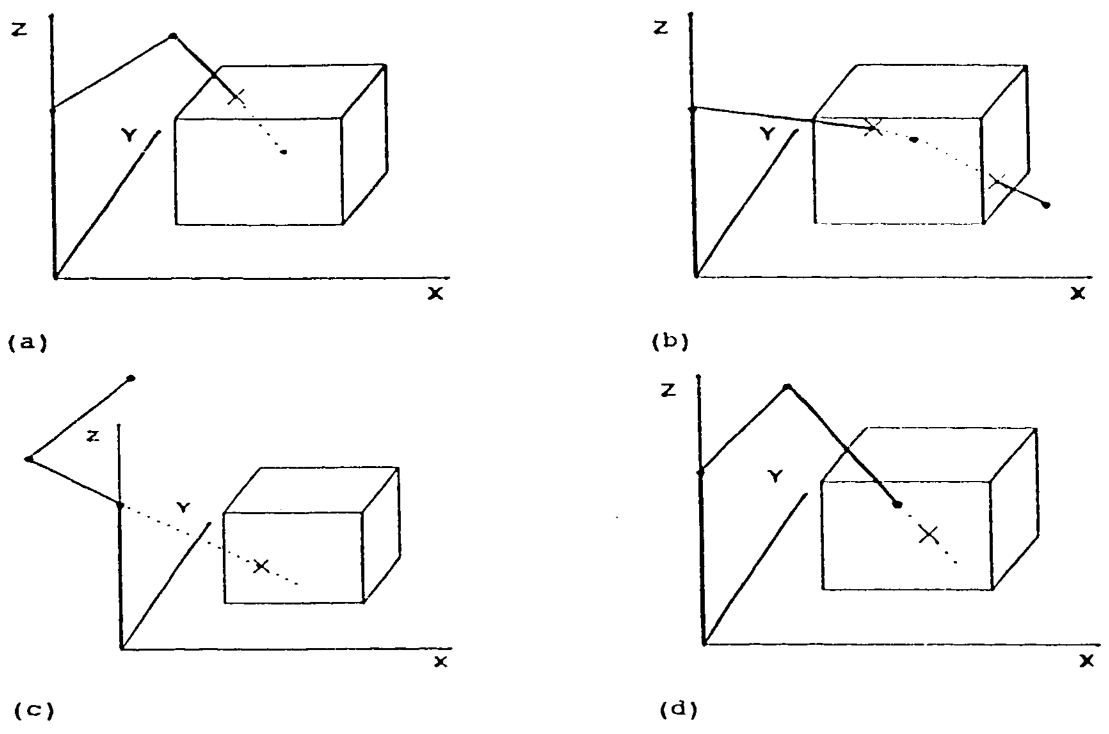
three equations simultaneously, the (X, Y, Z) coordinates of the intersection point between the infinite line and infinite plane can be found if it exists. A singular coefficient matrix (i.e. line and plane are parallel) signifies that no collision exists between the particular link and plane being tested.

Even if an intersection point is found, the algorithm must perform two additional checks before declaring the intersection point as a collision point. As a first check, the intersection point must fall within the dimensions of the obstacle box, and as a second check, the intersection point should lie between the end points of the colliding link. Figure 8 shows the possible collision and non-collision cases. For each combination of Θ_1, Θ_2 and Θ_3 joint angles entered, each robot link (upper and lower arm) must also be checked for collisions with each of the six planes that make up the obstacle box.

3.5 Visualization of the Navigation

Although visualization methods are subjective, they can provide insights into the behaviour of the planning algorithm. A visualization of the simulation gives a qualitative feel for the success of the learning algorithm and the emergent properties of the interactive schemas.

The GA-PATH produces a motor file that can be used with the robot animation program ANIMATE [Toogood, 1991] to display the path of the robot as sequence of points in world space. An example is shown in Figure 9.



(a),(b) Possible collision cases.

(c),(d) Possible non-collision cases.

Figure 8 Possible Collision and Non-collision Cases

(From Toogood and Wong, 1993)

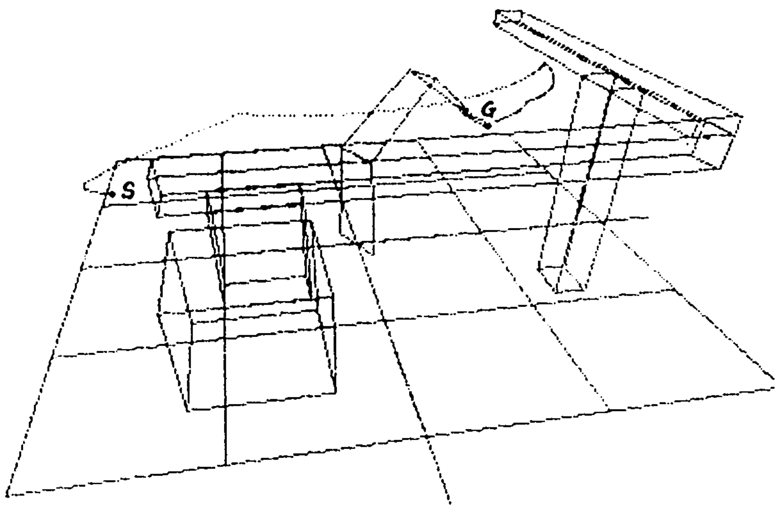


Figure 9. Simulated Robot Movement Using ANIMATE

4. GENETIC ALGORITHM FOR PATH PLANNING

4.1 Introduction

Genetic Algorithms are stochastic search techniques inspired by adaptation in evolving natural systems. GAs are well adapted to search for solutions in high dimensionality search space and are tolerant to the form of the function to be optimized.

As a search technique, a GA tries to maintain a balance between exploiting points in the search space that have already been reached and exploring other points that are yet to be tried. The reproduction operator exploits the "knowledge" present in the population by increasing the number of fitter individuals. The crossover operator explores the search space by producing new points to evaluate. This simultaneous exploration and exploitation moves the search toward the neighbourhood of optimal solutions. The convergence time and solution quality depend on both the nature of the problem and the GA parameters (population size, string length, probabilities of selection, crossover, etc.).

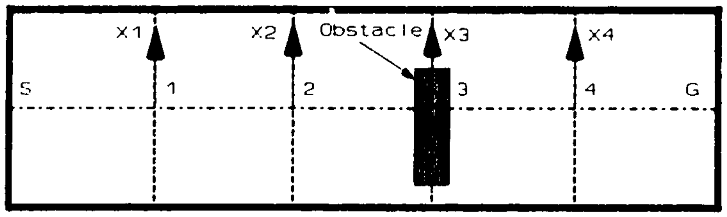
Since the robot path planning is a huge space search process, the exploration and exploitation nature of the GAs can be adopted to perform the search and determine optimal trajectory parameters.

In this chapter, a GA-PATH program is described that is used to generate collision-free paths in joint space. The algorithm used in the GA-PATH for trajectory generation is described in detail. The ultimate goal here is to design a path planner which can generate a true global optimal trajectory. The movement of the first three joints which provide the basic positioning capability of the Mitsubishi RM101 robot is considered. The robot is considered the only moving object in the work cell, and all the obstacles in the workspace are stationary. Collision-free paths from a start point to a goal point in the work cell are generated by the GA-PATH.

4.2 Experimental Tests

4.2.1 1-D Case

Encoding Scheme:



Coding for 1-D case: Trajectory = S { X_1,X_2,X_3,X_4 } G

Figure 10. 1-D case

The basic trajectory generation strategy starts with setting up a number of node points along the Start-Goal line (Figures 10). Then, lines are constructed perpendicular to the Start-Goal line at each node point in the search space. A local coordinate system is then established at each node. Therefore, the GA-PATH generated knot points can be located along the lines by discretizing them. The final trajectory will compose one knot point from each line with the start and goal point. The number of the node points depend on the complexity of the workspace. The more complex the workspace, the more node points are suggested due to the nature of joint interpolated control scheme of RM101 Robot, but the criteria is to encode as few variables as possible to reduce the memory/speed requirements.

Fitness Evaluation:

Evaluation functions in GA applications are external to the GA. The evaluation of trajectories could be based on either joint angle movements minimization along the path, or distance minimization (i.e. find the shortest path), or some other indices.

The output from the evaluation function is a positive number indicating the relative fitness of a decoded structure to the other members of the population. The performance criteria were designed so that the end-effector position must be collision free, and at the same time with the minimized total distance. The fitness value was therefore assigned to achieve a balance among these criteria.

(1) Minimize the total distance: the total distance was calculated by

$$\text{Error}_1 = \sum \text{ABS}\{600\text{-SQR}[(P_{x,i} - P_{x,i-1})^2 + (P_{y,i} - P_{i,i-1})^2]\}$$

where i = 0 signifies Start Point and i = N+1 signifies Goal Point. P_x, P_y are the X and Y coordinates of a knot point.

(2) Obstacle avoidance: collisions were checked at each node point along a trajectory, If there is a collision, a penalty will be added to the fitness, otherwise, the fitness remain unchanged.

$$\text{Error}_2 = \sum \text{ABS}\{600\text{-SQR}[(P_{x,i} - P_{x,i-1})^2 + (P_{y,i} - P_{i,i-1})^2]\} \qquad \text{Collision occurs}$$

$$\text{Error}_2 = 0 \qquad \text{No collisions}$$

(3) The total Error will be

$$\text{Error} = K_1 * \text{Error}_1 + K_2 * \text{Error}_2$$

Where, the K's are weighting factors.

The fitness of a trajectory is a relative measure which depends on the performance of other members of the population. The fitness is a positive monotone transformation of error designed to give an appreciative relative reward to different members of the population. An exponential scaling is used so that,

$$\text{Fitness} = \text{Exp}^{(-\text{Error}/\kappa)}$$

where κ is a constant used to adjust the fitness values in a reasonable range.

Reproduction and Crossover Strategy:

Roulette wheel selection mechanism was employed in this 1-D case test. Due to the encoding scheme of the trajectory representation, each individual string will have N*M long binary digits in the case of N node points and M-bit binary digits encoding each of the X_i's coordinate. The defining length of the string will increase with the number of node point increases. As described in Chapter 2, the disruptive potential of uniform crossover does not depend on the defining length of hyperplanes, in another words, the effectiveness of uniform crossover is not sensitive to the defining length of a chromosome.

By comparing with many other crossover techniques, uniform crossover appears to have three potentially important virtues: (1) the disruption of hyperplane sampling does not depend on the defining length of the hyperplane. (2) the disruption potential is easily controlled via a single parameter which is the uniform crossover probability; (3) when a disruption does occur, uniform crossover results in a minimally biased exploration of the space being searched [Spears and De Jong, 1991]. Therefore, uniform crossover was chosen as the crossover operator. Standard mutation was chosen to provide a small amount of random search, and helps ensure that no point in the search space has a zero probability of being examined.

Test Parameters and Weighting Factors:

There are three parameters which have significant effects on the performance of the Genetic Algorithm. These parameters are: population size (n), crossover probability (p_c), and mutation probability (p_m). In De Jong's [1975] and Grefenstette's [1986] study of Genetic Algorithms in function optimization and optimization of control parameters for Genetic Algorithms, a series of parametric studies across a broad spectrum of problems suggested that good GA performance requires the choice of a higher crossover probability, a low mutation rate (inverse proportional to the population size), and a moderate population size.

Based on the previous studies, the following parameters were adopted for the evaluation of the GA-PATH algorithm. The convergence criterion was set such that either the GA process reaches the maximum generation or the average fitness value comes to certain range.

Population size (n) = 20; 50

Chromosome length (M) = 5 bits (for each coded parameter)

Maximum generation = 100

Probability of crossover (p_c) = 0.1, 0.4, 0.9

Probability of mutation (p_m) = 0.002

The weighting factors were set as: $K_1 = 1$, $K_2 = 20$ and $K = 500$.

Results and Discussion

A examination was set up at first to see whether the GA-PATH works at all. The GA-PATH was assigned to optimize a simple path. The search space was a 600 by 160 (Length by width) units rectangular plane (Figure 10). There were four node points (N=4) between starting point and goal point. At each node point the position on a line perpendicular to the Start-Goal line was discretized using a 5-bit (M=5) binary string. The binary string assigned to each node point were concatenated to form a 5*4=20 bits binary string. One obstacle (90 units width) at the third node point was presented.

Figures 11 to 14 show the simulation results from six experiments in the 1-D case. It was observed that the GA-PATH algorithm performed very well in 1-D case trials. Figures 11-12 illustrated that the 0.4 crossover probability gave the best result when the population size was set to 50. Both lower (0.1) and higher (0.9) crossover probabilities took less generations to converge, but the results were not as good as (in terms of distance) that 0.4 probability crossover produced. The evolution process converged much faster with the small population size (n=20) than the larger ones (n=50). As shown in Figure 13 and 14, all trials were able to find collision free paths.

The simulation results demonstrated that the population size and P_c values both have significant effects on the convergence rate of GAs. The premature convergence cases caused by the extreme P_c values (0.1 or 0.9) were also observed in some of the trials. The moderate P_c value (0.4) for the uniform crossover kept a better balance between the exploration and exploitation search process.

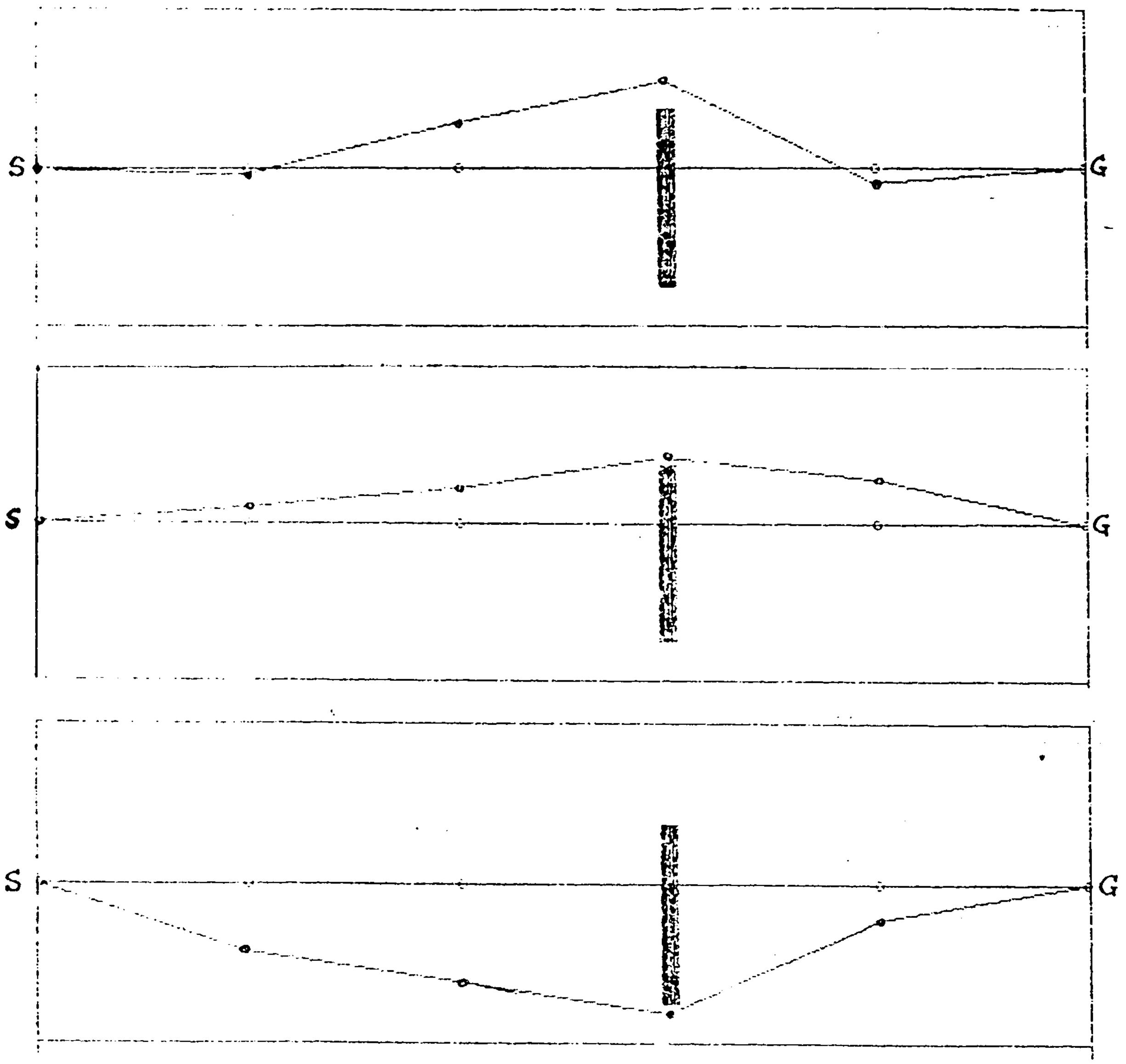


Figure 11. GA-PATH for 1-D Case (n=50)

[P_c=0.1 (Top), 0.4 (Middle), 0.9 (Bottom)]

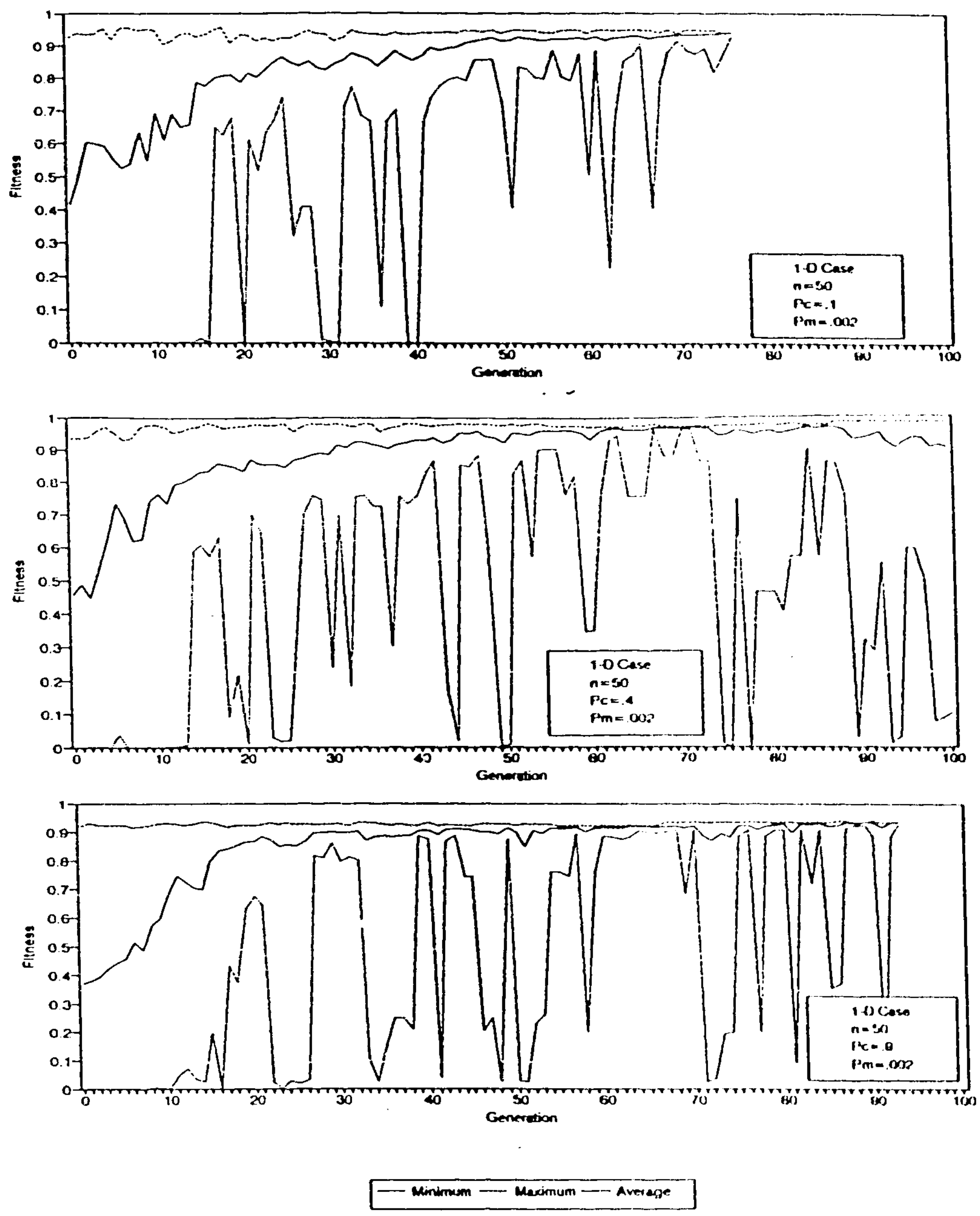


Figure 12. GA-PATH Statistics for 1-D Case (n=50)

[P_c=0.1(Top), 0.4 (Middle), 0.9 (Bottom)]

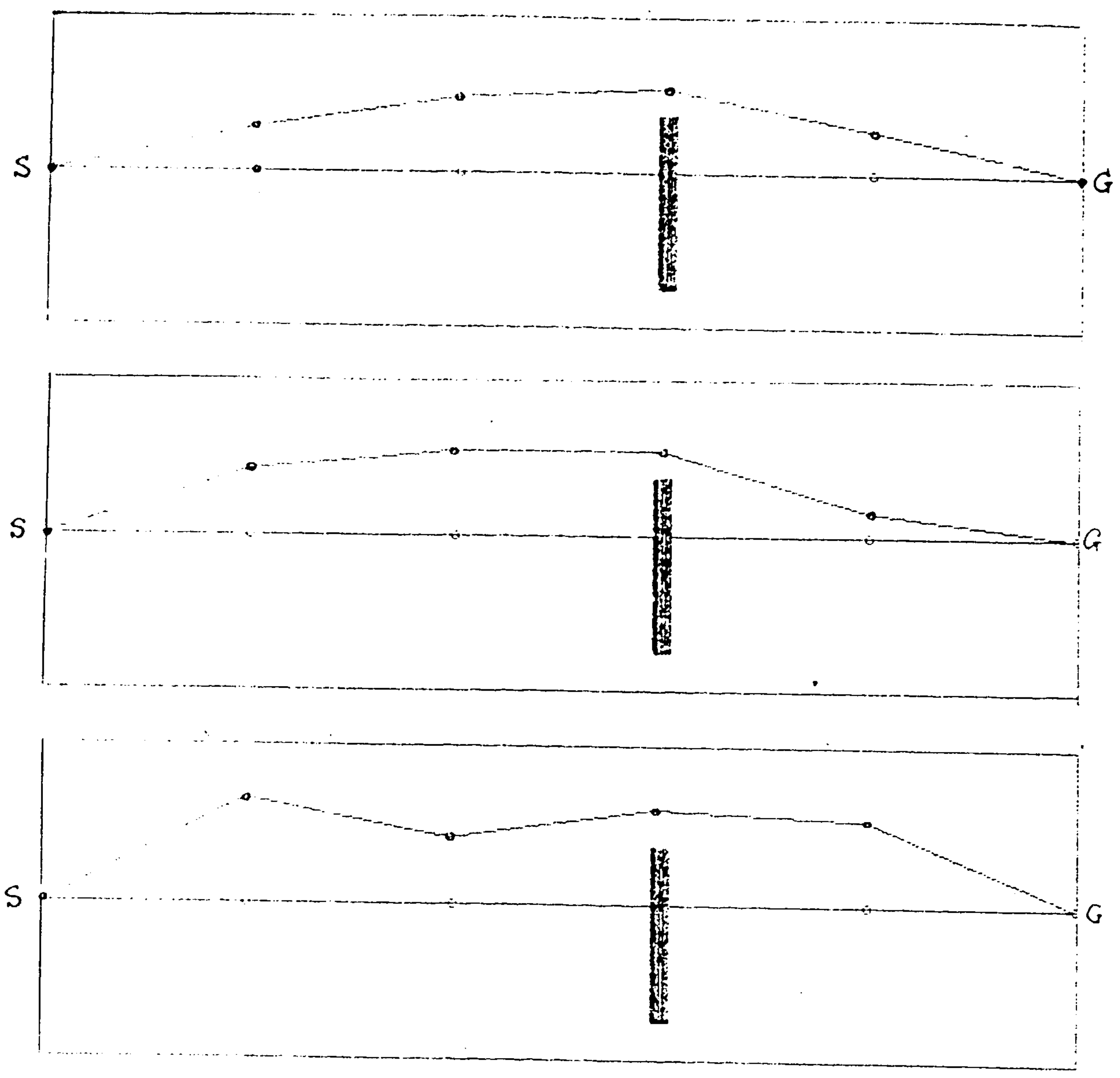


Figure 13. GA-PATH for 1-D Case (n=20)

[P_c=0.1 (Top), 0.4 (Middle), 0.9 (Bottom)]

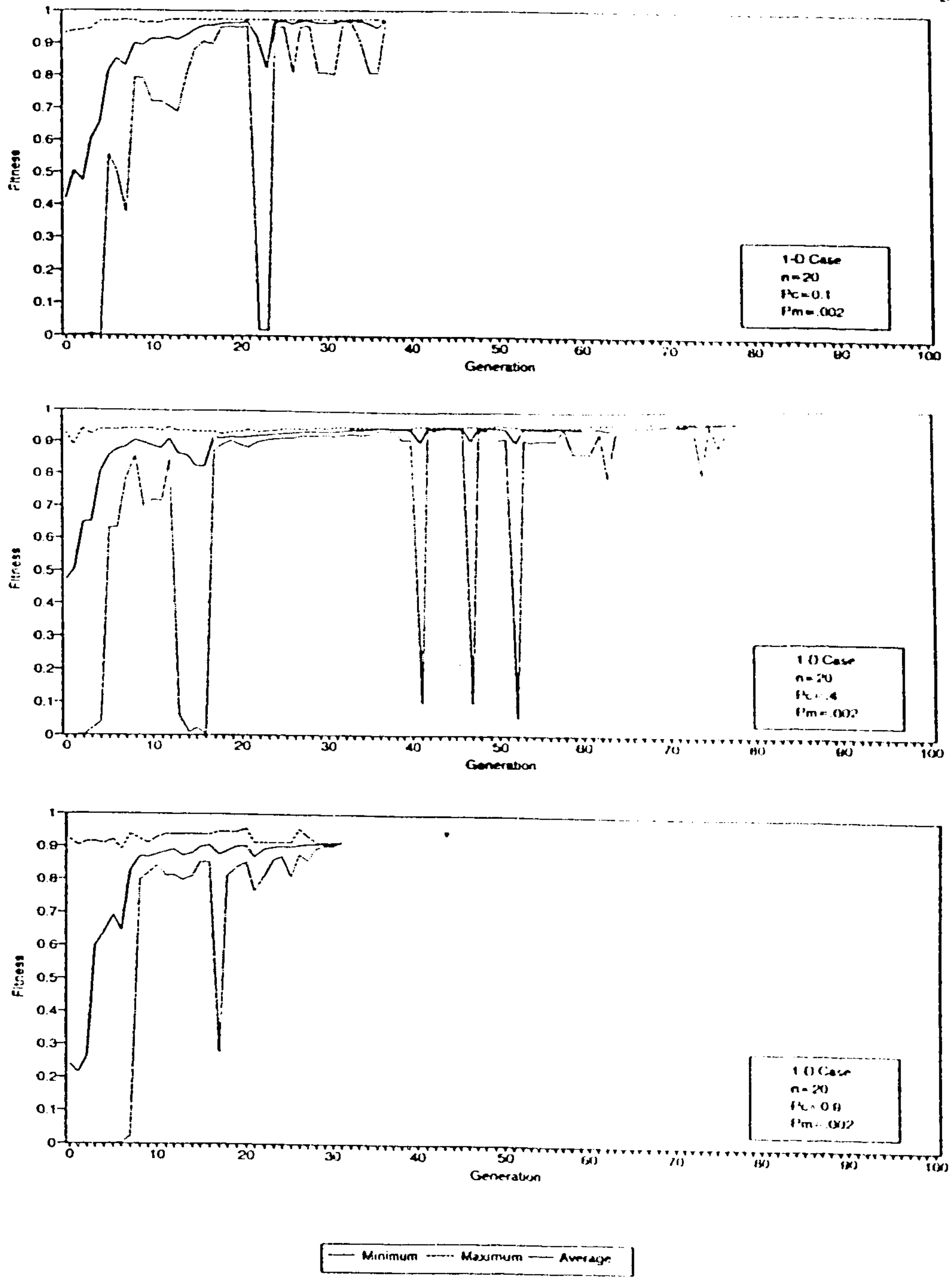
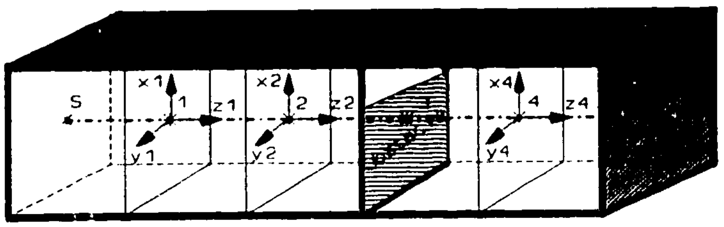


Figure 14. GA-PATH Statistics for 1-D Case (n=20)

[P_c=0.1 (Top), 0.4 (Middle), 0.9 (Bottom)]

4.2.2 2-D Case

Encoding Scheme:



Coding for 2-D case: Trajectory = S { $(X_1,Y_1),(X_2,Y_2),(X_3,Y_3),(X_4,Y_4)$ } G

Figure 15. 2-D case

The basic trajectory generation strategy is the same as in 1-D case. It starts with setting up a number of node points along the Start-Goal line (Figures 15). Then, planes are constructed perpendicular to the Start-Goal line at each node point in the search space. A local coordinate system is then established at each node. Therefore, the GA-PATH generated knot points can be located along the plane by discretizing them. The final trajectory will compose one knot point from each plane with the start and goal point.

Fitness Evaluation:

The evaluation of trajectories was based on joint angle movements minimization along the path. The performance criteria were designed so that the end-effector position must be collision free, and at the same time with the minimized total displacement X_i and Y_i. The fitness value was assigned as

(1) Minimize the X_i's and Y_i's displacement: the displacement was calculated by

$$\text{Error}_1 = \sum\{(P_{x,i} - P_{x,i-1})^2 + (P_{y,i} - P_{i,i-1})^2\}$$

where i = 0 signifies Start Point and i = N+1 signifies Goal Point. P_x, P_y are the X and Y coordinates of a knot point.

(2) Obstacle avoidance: collisions were checked at each node point along a trajectory, If there is a collision, a penalty will be added to the fitness, otherwise, the fitness remain unchanged.

$$\text{Error}_2 = \sum\{(P_{x,i} - P_{x,i-1})^2 + (P_{y,i} - P_{i,i-1})^2\} \qquad \text{collision occurs}$$

$$\text{Error}_2 = 0 \qquad \text{No collisions}$$

(3) The total Error will be

$$\text{Error} = K_1 * \text{Error}_1 + K_2 * \text{Error}_2$$

Where, the K's are weighting factors.

An exponential scaling is used so that,

$$\text{Fitness} = \text{Exp}^{(-\text{Error}/k)}$$

where K is a constant used to adjust the fitness values in a reasonable range.

Reproduction and Crossover Strategy:

Due to the increased complexity of the search space, in order to minimize the stochastic errors associated with the roulette wheel selection, remainder sampling without replacement and uniform crossover operator were chosen for the reproduction mechanism. Based on the encoding scheme of the trajectory representation, each individual string will have 2*N*M long binary digits in the case of N node points and M-bit binary digits encoding each of the X and Y coordinates. Uniform crossover and standard mutation were chosen for the 2-D case.

Test Parameters and Weighting Factors:

The following parameters were adopted for the evaluation of the GA-PATH algorithm in 2-D case. The convergence criterion was set the same as in 1-D case.

Population size (n) = 20; 50

Chromosome length (M) = 6 bits (for each coded parameter)

Maximum generation = 100

Probability of crossover (p_c) = 0.1, 0.4, 0.9

Probability of mutation (p_m) = 0.002

The weighting factors were set as: $K_1 = 1/1000$, $K_2 = 10$ and $K = 100$.

Results and Discussion

In the 2-D case, a 600*150*120 (Length by width by height) unit spatial tunnel was searched for feasible paths. Figure 15 shows that one obstacle (crossection is

150*85) was located in the search space at the 3rd node. At each one of the four node points (N=4), a crossection plane perpendicular to the Start-Goal line was applied. If the Start-Goal line was assigned as Z direction, then the local X, Y, coordinates were imposed at the intersection point between the line and the plane. The parameters X and Y were each coded by a 6-bit (M=6)binary string. Therefore, a concatenated 2*6*4=48 bit string was formed for each individual in the population. The GA parameters were set the same as in 1-D case. Figures 16-25 show the GA-PATH simulation results for the 2-D case.

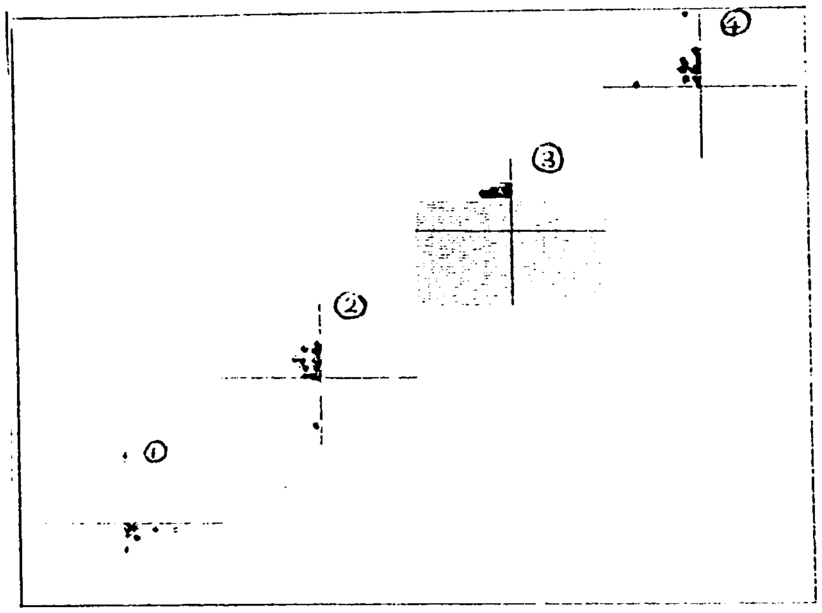
It was observed that the algorithm successfully generated collision free trajectories, and the rate in which improvements are introduced is impressive. It was also observed that in general the algorithm performed very well, and the generated trajectories were very close to the optimal one (in terms of minimized joint angle displacement).

The simulation results illustrated that all GA parameters being tested performed similarly well for the "so far" best trajectory generated. The fitness function values show little diversity, and the "best" fitness occurs (Figure 19,23) quite early in the evolution process. There were no significant differences between the performances of population size 50 (Figure 16-19) and 20 (Figure 20-23), this demonstrated that uniform crossover handled relatively small population size very well. For larger population size (n=50), different P_c's had no significant effect on the GA-PATH performance. For small population size (n=20), the effect on the GA-PATH convergence rate was observed due to the different P_c's. The simulations demonstrated the exploration and exploitation power of the uniform crossover operator. Moderate uniform crossover probability was observed that could provide better performance. Figure 24 and 25 show ano'her trial with $P_c=0.5$.

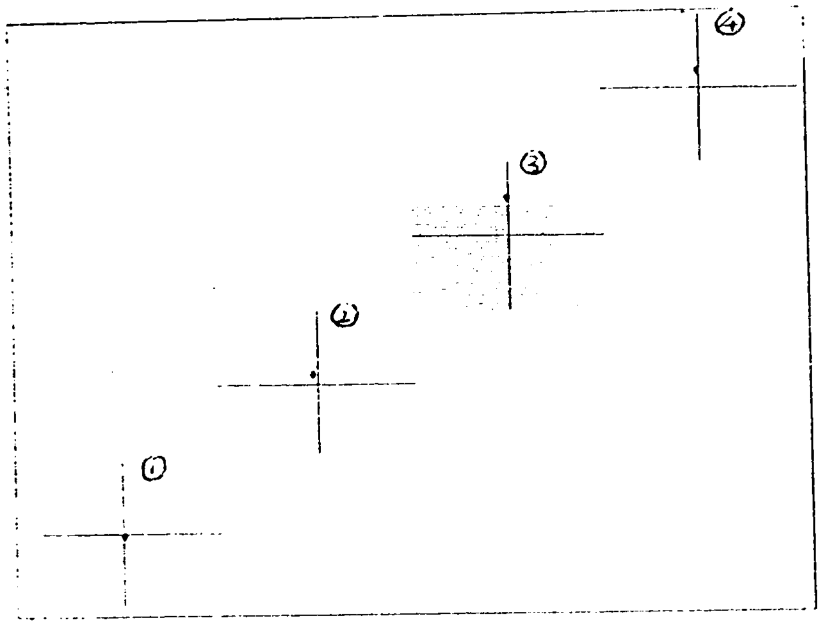
Although the best trajectories at the end of each experiment are different, their performance is of a similar quality. This phenomenon indicates that the GA-PATH converges to different regions in the trajectory space, this is an example of a flat landscape search space. The stopping criterion (such as generation size) could also force the evolution process to terminate at the early stages.

It should be noted that a trajectory which has a similar fitness has no advantage over other trajectories of a similar fitness and therefore, does not experience selection pressure. It was suggested by Davidor [1991] that often one should be satisfied with one good solution, and not be too concerned about other aspects of the space, including the possibility of even better solutions, provided of course, that the solution in hand is good enough. It does not guarantee "best" solution in the sense of global optimal, but only "better" solutions than what was started with.

The 1-D and 2-D case trials demonstrated the searching power of the GA-PATH. With moderate crossover probability, the uniform crossover operator could be very effective for keeping balances between the exploration and exploitation search in searching complex spaces.

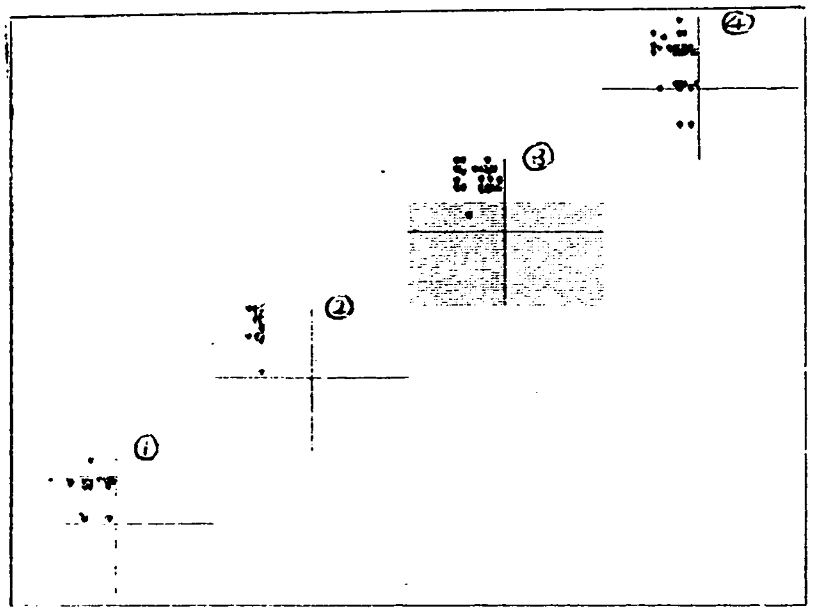


a) Whole population

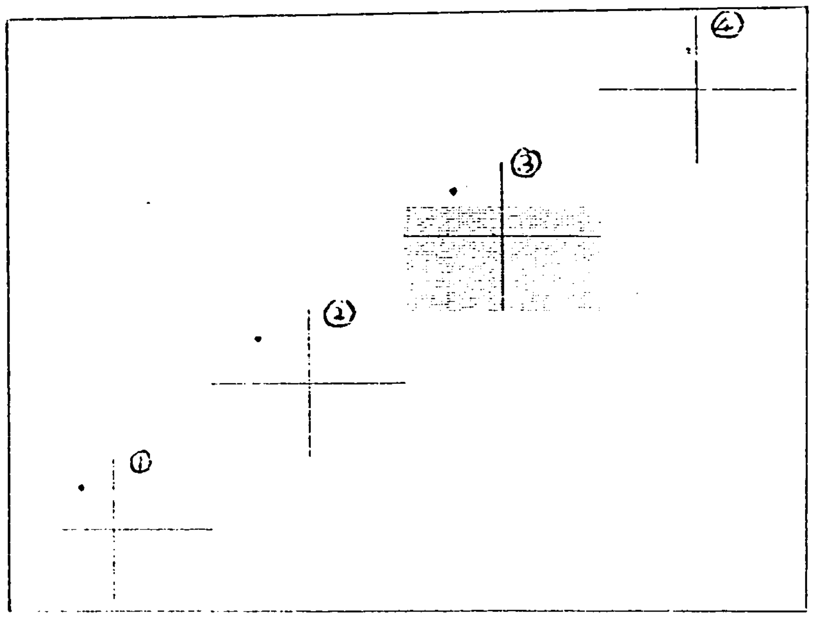


b) Best string

Figure 16. GA-PATH for 2-D Case (n=50, P_c=0.4)

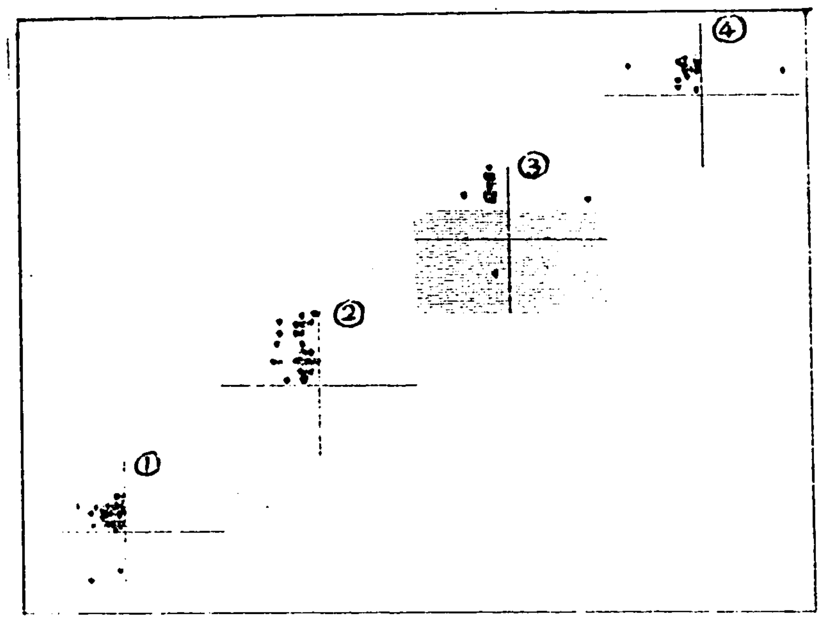


a) Whole population

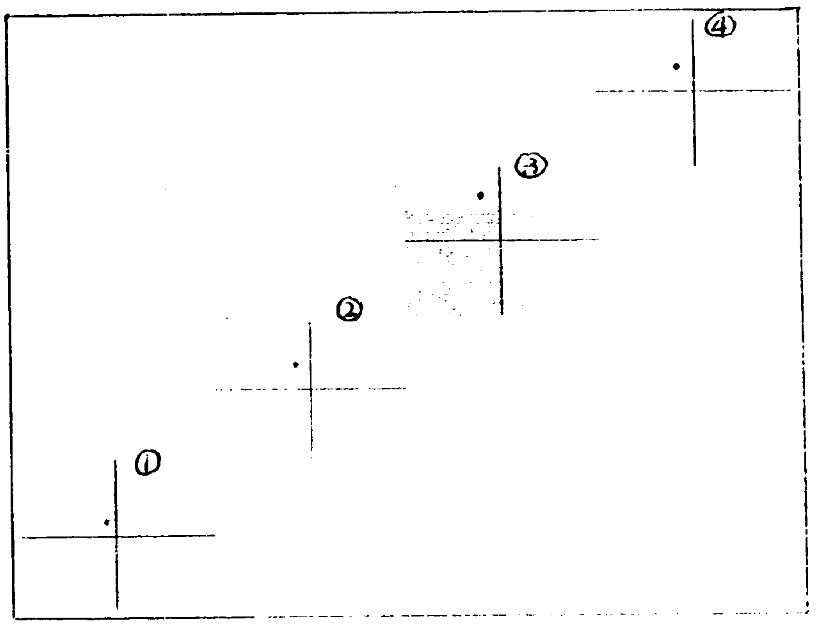


b) Best string

Figure 17. GA-PATH for 2-D Case (n=50, P_c=0.1)



a) Whole population



b) Best string

Figure 18. GA-PATH for 2-D Case ($n=50$, $P_c=0.9$)

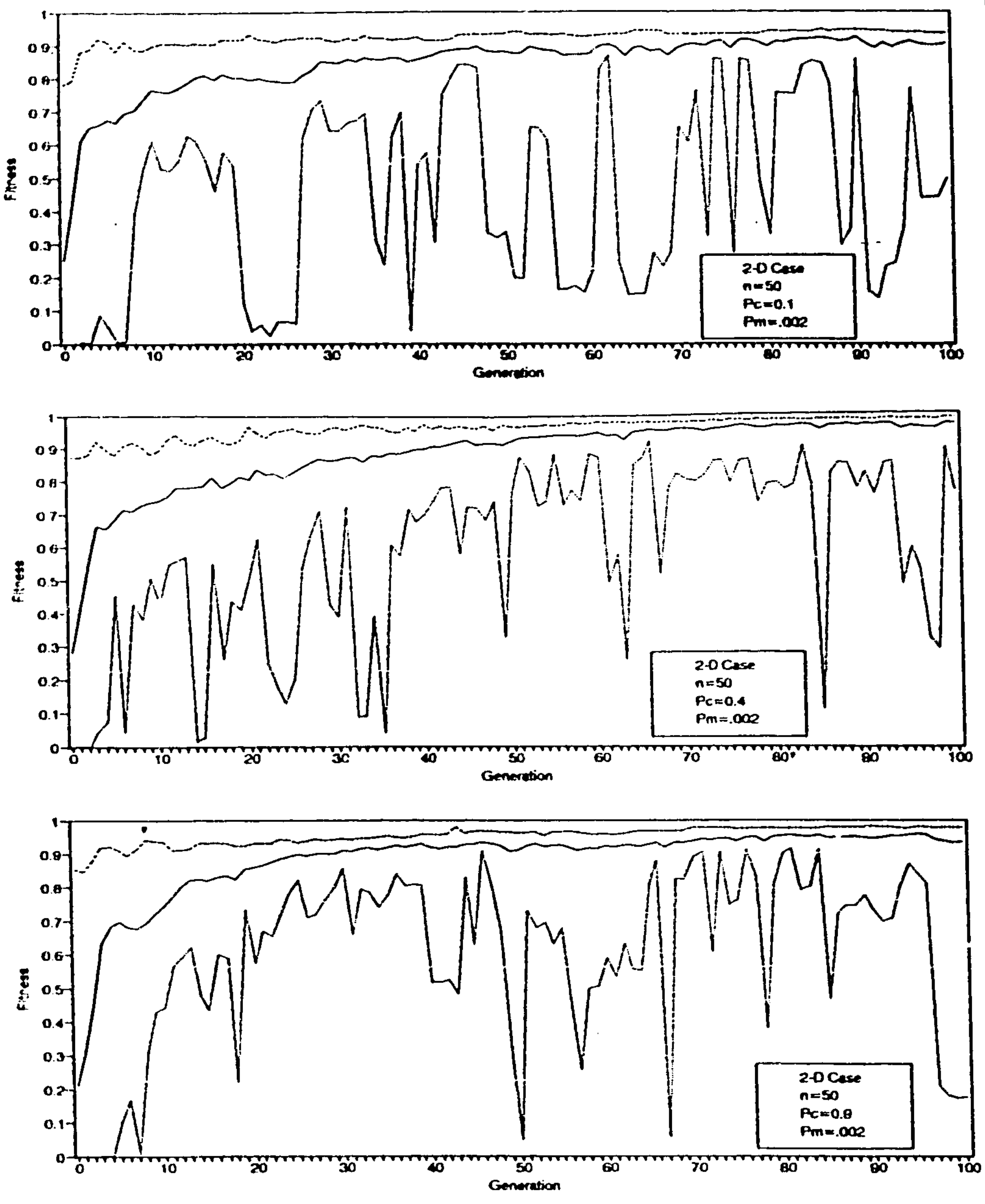
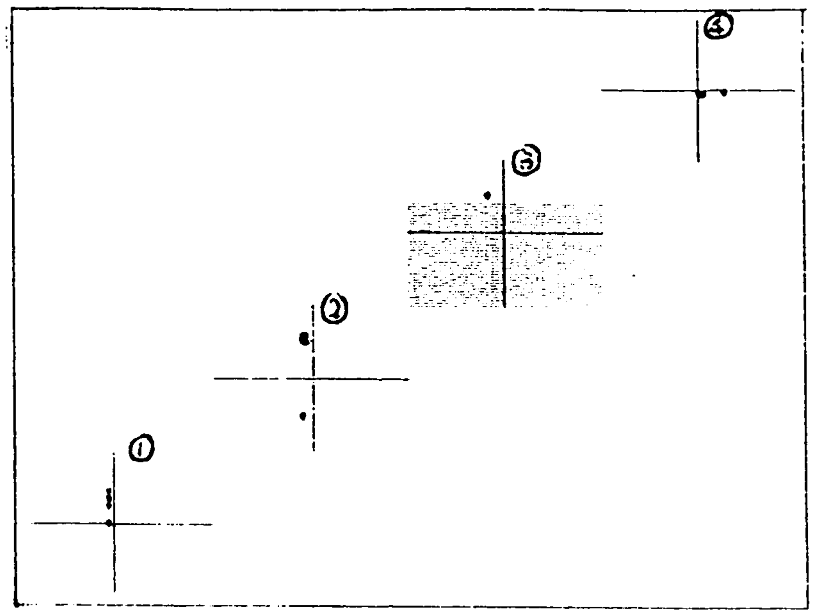
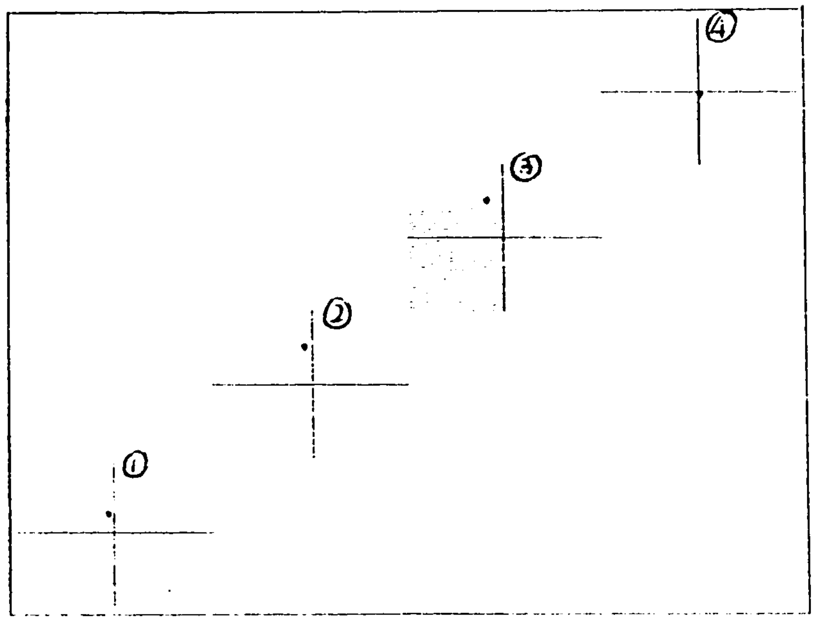


Figure 19. GA-PATH Statistics for 2-D Case (n=50)

[P_c=0.4 (Top), 0.1 (Middle), 0.9 (Bottom)]

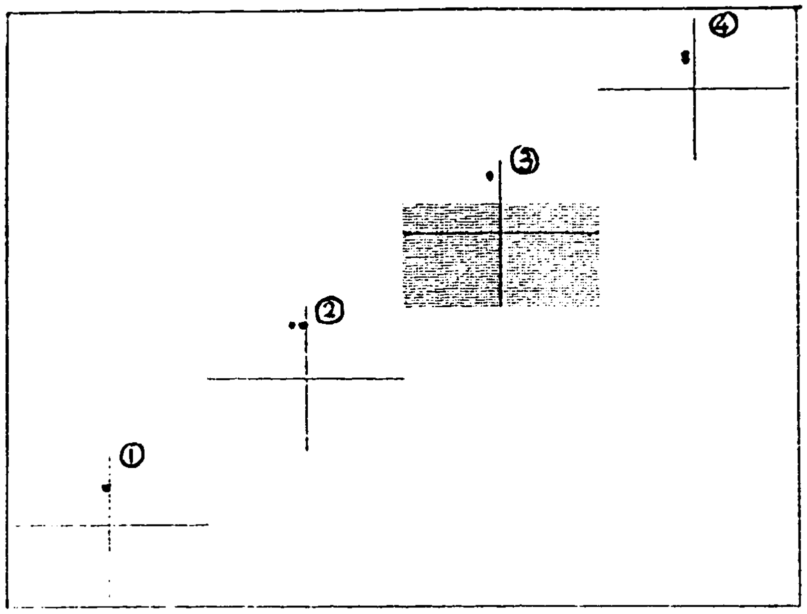


a) Whole population

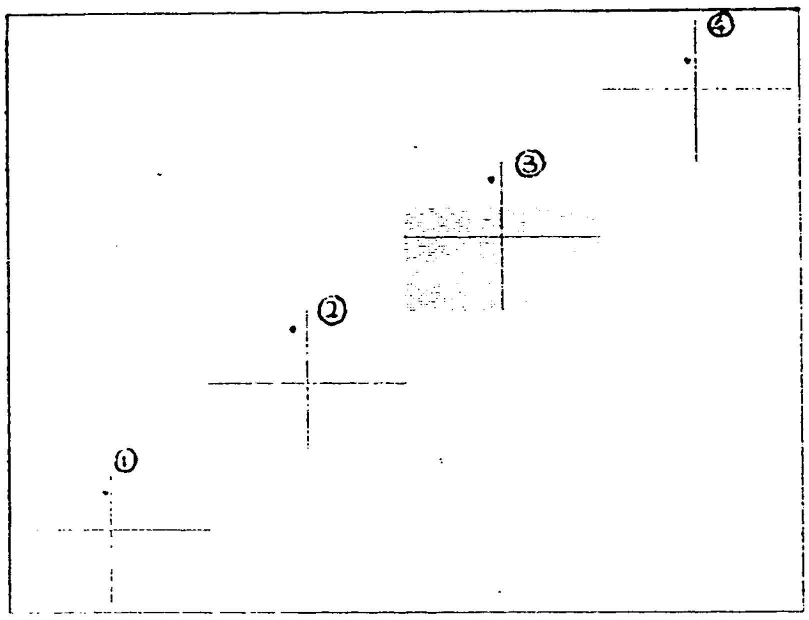


b) Best string

Figure 20. GA-PATH for 2-D Case (n=20, P_c=0.4)

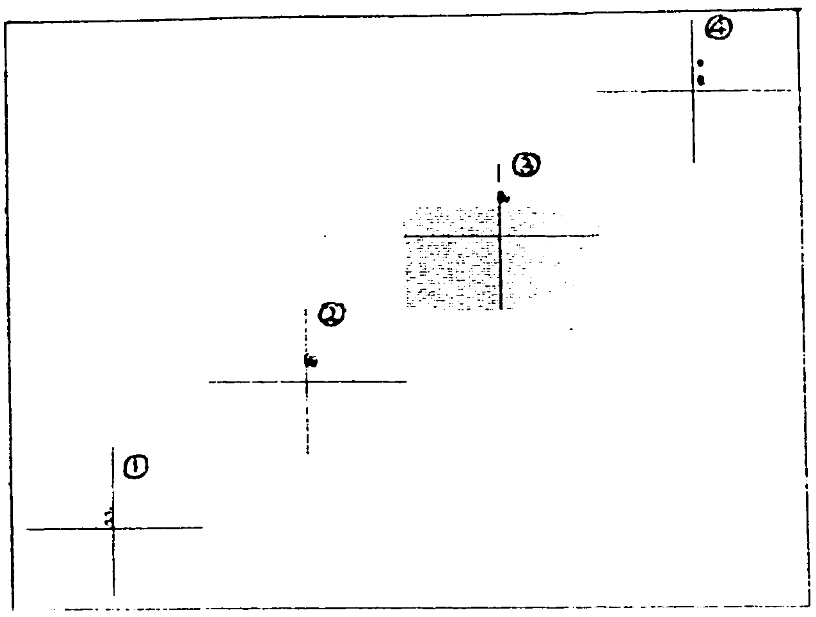


a) Whole population

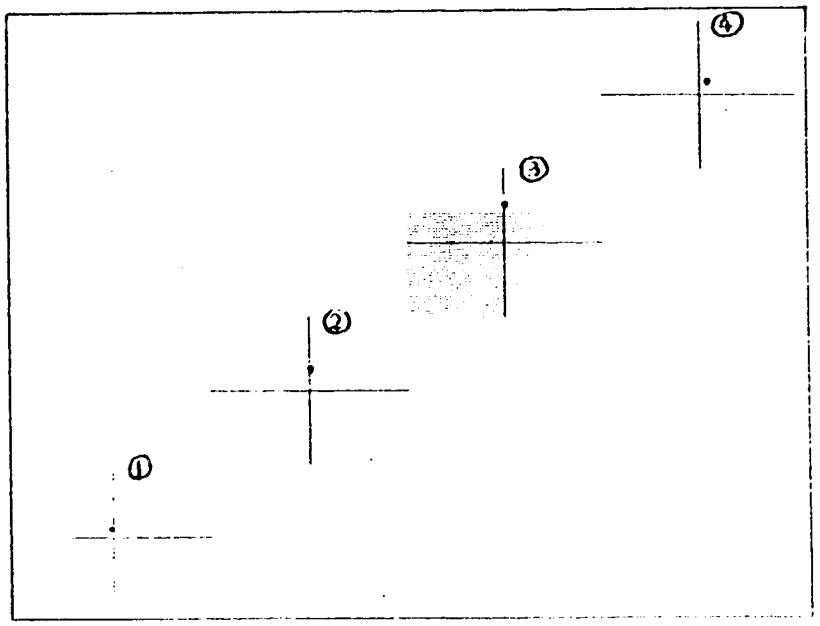


b) Best string

Figure 21. GA-PATH for 2-D Case ($n=20$, $P_c=0.1$)



a) Whole population



b) Best string

Figure 22. GA-PATH for 2-D Case (n=20, P_c=0.9)

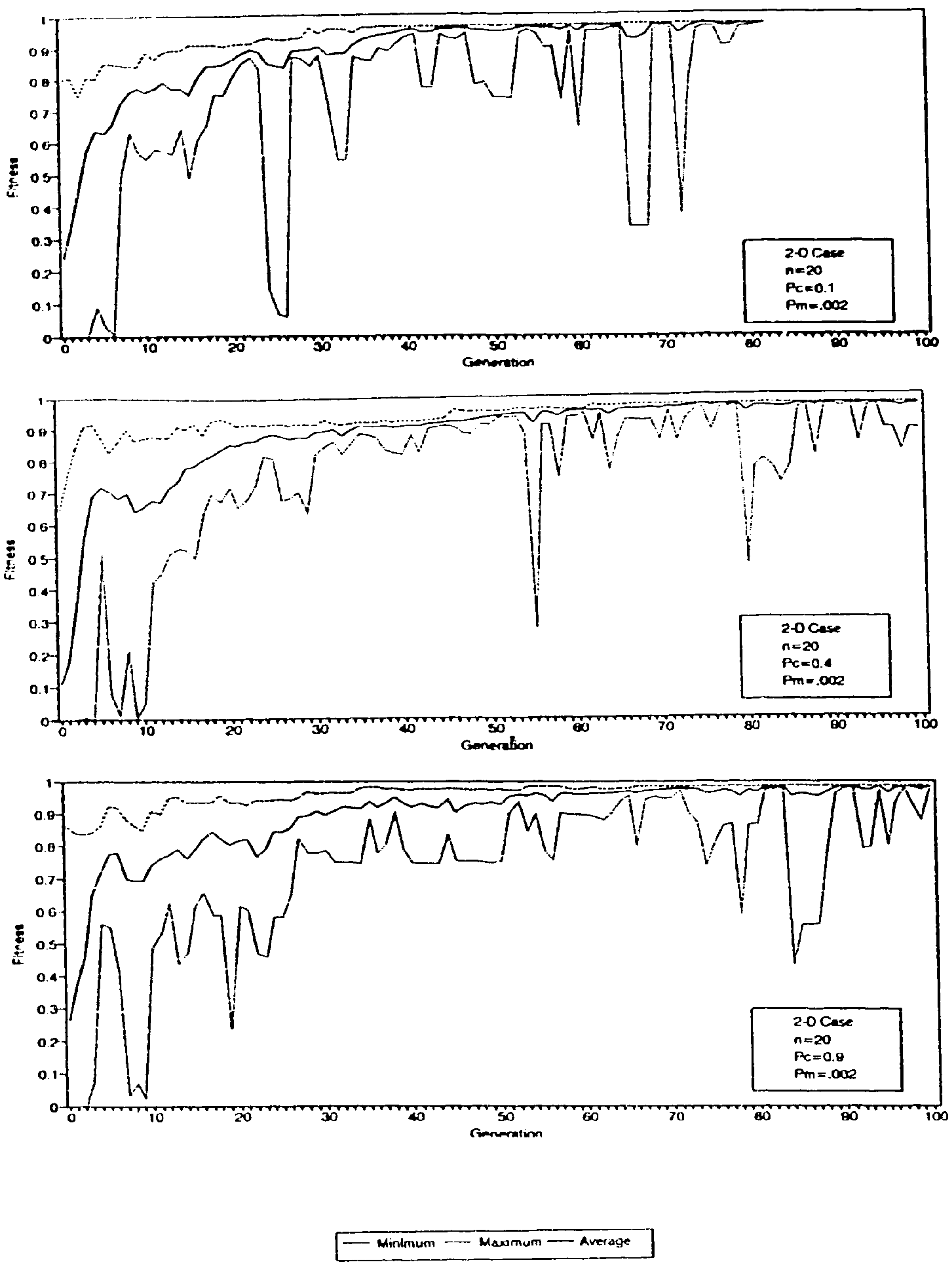
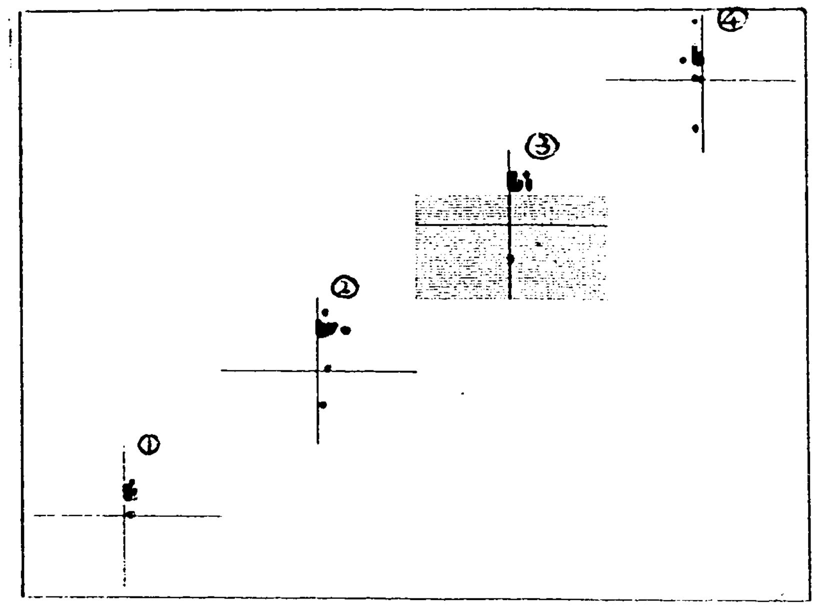
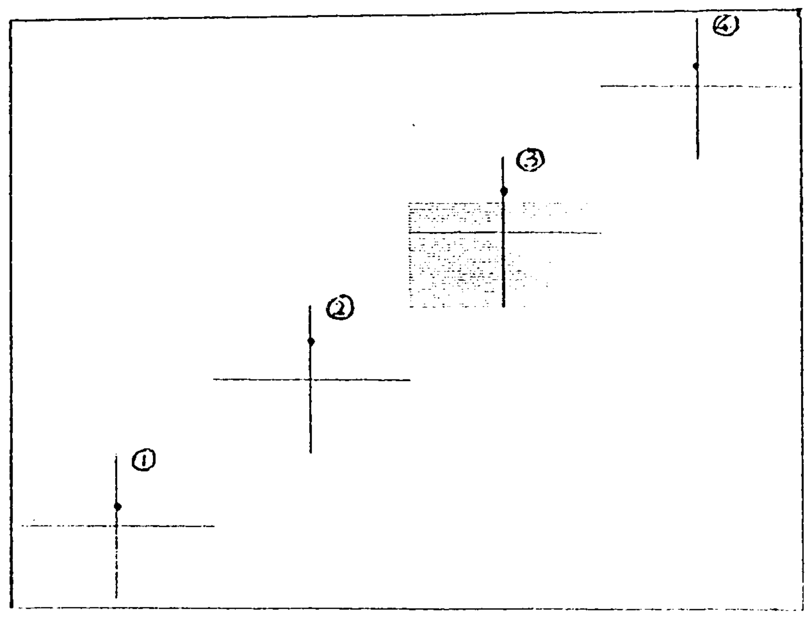


Figure 23. GA-PATH Statistics for 2-D Case (n=20)

[P_c=0.1 (Top), 0.4 (Middle), 0.9 (Bottom)]



a) Whole population



b) Best string

Figure 24. GA-PATH for 2-D Case (n=50, P_c=0.5)

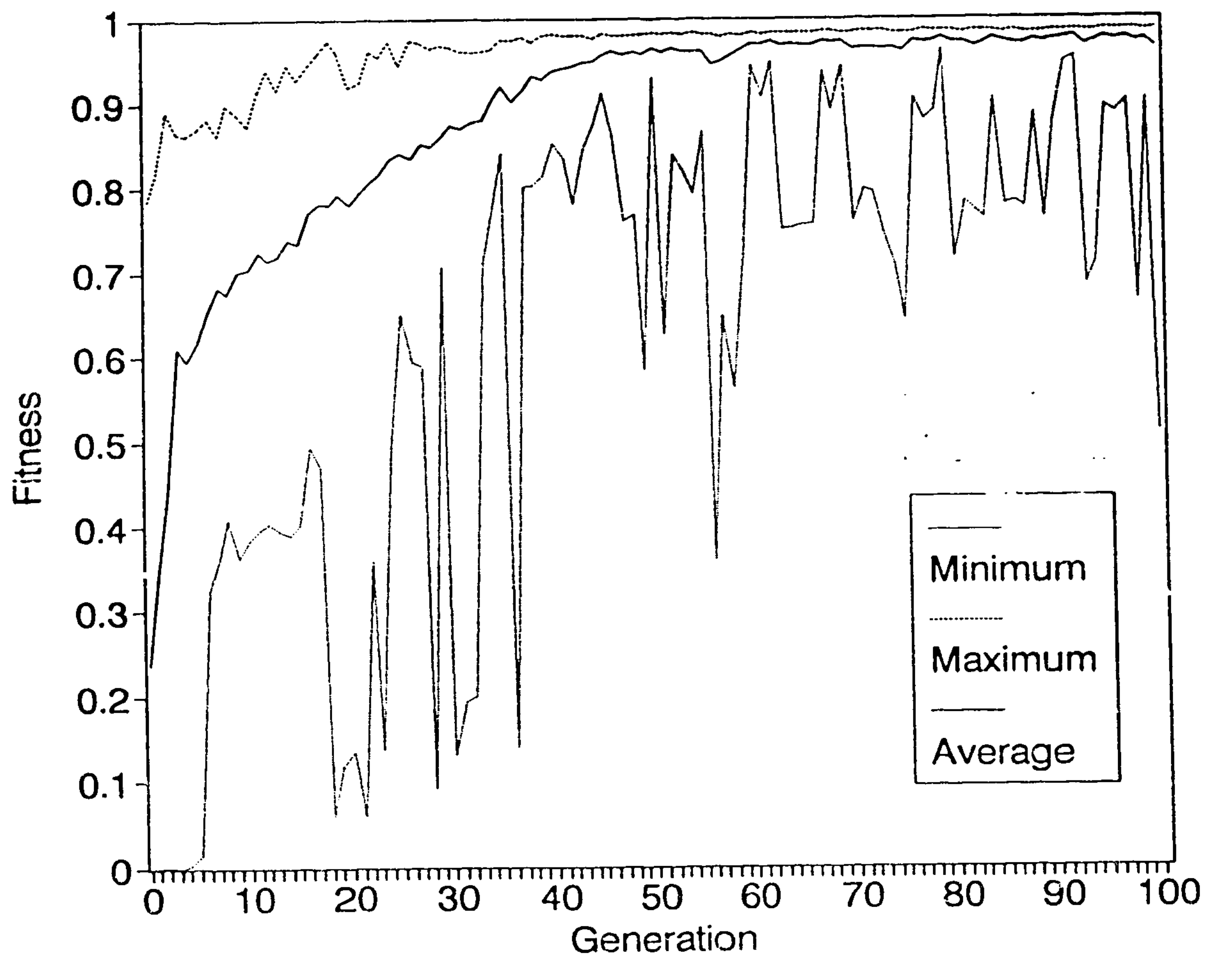


Figure 25. GA-PATH Statistics for 2-D Case (n=50, P_c=0.5)

4.3 3-D Case: GA-PATH for Mitsubishi RM101 Robot

Based on the promising results obtained from 1-D and 2-D cases, the GA-PATH was applied to Mitsubishi RM101 robot for collision-free trajectory generation in a work space occupied with stationary obstacles. To plan the path among a set of obstacles, a joint space description of the work cell obstacles was utilized. The use of configuration space transforms the problem of planning the motion of a complex dimensional object in cartesian space into the problem of planning the path of a point in joint space. The term C-obstacle was used for the set of points in configuration space for which any part of the robot collides with an object in the work cell. When working in cartesian space, multiple obstacles are usually treated as separate entities. However, it is quite common for multiple C-obstacles to have overlapping volumes in joint space, a result of the robot being in collision with several obstacles at the same instant.

The trajectory of a robot from a start to a goal point can follow many possible paths, depending on the control system used. The Mitsubishi RM101 uses a joint interpolated control scheme. This coordinated motion results in the robot path following a straight line in joint space when directed to move to another pose. This greatly simplifies the planning problem since the exact path of the robot between any two poses can be precisely defined. This is not the case with other control schemes (eg. independent joint PID control).

In order to generate a collision-free path from a start point to a goal point among several C-obstacles in configuration space, the GA-PATH algorithm was devised as follows:

(1) Set start point and goal point in joint space. Calculate α and β which represent the orientation of the Start-Goal line in joint space (see Figure 26).

(2) Initialize the parameters for the Genetic Algorithm. This includes population size, maximum number of generation, chromosome size for each parameter code, boundary conditions for generated parameters, probabilities for crossover and mutation.

(3) Set node points along the straight line between the start and goal points. The number of node points depends on the complexity of the workspace. The maximum number of node points for the Mitsubishi RM101 Robot path generation was set to 12. As a rule of thumb, we want to encode as few variables as possible to reduce the memory/speed requirements.

(4) Define a plane at each node point. At each node point, a plane perpendicular to the Start-Goal line was created in joint space. A local X, Y, and Z coordinate system was then defined with the origin located at the node point, and the Z coordinate was defined along the line pointing toward the goal. The X and Y coordinates follow the right hand rules. Thus, all three angles $(\Theta_1,\Theta_2,\Theta_3)$ at node i are able to be defined by any point generated on the perpendicular plane through the mapping $(X_i, Y_i, Z_i) \rightarrow (\Theta_1,\Theta_2,\Theta_3)_i$, or, since the Z_i's are fixed, we can write this mapping as $(X_i, Y_i) \rightarrow (\Theta_1,\Theta_2,\Theta_3)_i$. Note that this reduces the number of variables required to describe each knot point from 3 to 2.

(5) Generate node points at local coordinate system. For every local coordinate system i, a Z_i value is calculated based on the location of the *ith* node point, and the X_i and Y_i coordinates are randomly generated within certain boundaries by a GA. The parameters

X and Y are coded as a M-bit binary each. It was assumed that X and Y are bounded in the closed intervals [X_{min}, X_{max}] and [Y_{min}, Y_{min}] respectively. Parameters X and Y are then coded into a 2*N*M binary string as:

$$\text{GA Coding: Trajectory} = S \{ (X_1, Y_1), (X_2, Y_2), \ldots (X_N, Y_N) \} G$$
$$= S \{ (\Theta_1,\Theta_2,\Theta_3)_1, (\Theta_1,\Theta_2,\Theta_3)_2, \ldots (\Theta_1,\Theta_2,\Theta_3)_N \} G$$

Because of the coding scheme, knot points cannot be revisited as in the work of Shibata (1993). Furthermore, unlike the work of Khoogar et al. (in which the GA can terminate with the end-effector a considerable distance away from the goal point), in GA-PATH, the goal point is explicitly included in the path.

In the decoding procedure, the binary string is transformed into the corresponding parameters X_i and Y_i according to their boundary settings. The translation formula can be expressed as:

$$X_i = X_{min} + (X_{max} - X_{min}) * X_b/(2^M - 1)$$
$$Y_i = Y_{min} + (Y_{max} - Y_{min}) * Y_b/(2^M - 1)$$

where X_{min} and X_{max}, Y_{min} and Y_{max} are the minimum and maximum values of X and Y respectively. X_b and Y_b are the actual integer value of the binary string of the corresponding parameters.

(6) Coordinate system transformation and joint angles calculation for knot points. Post multiplication was carried out to perform the relative transformations from the local cartesian coordinate system (X_i, Y_i, Z_i) to configuration space (Θ_1,Θ_2,Θ_3). Figure 26

shows the transformation from world space to joint space. The coordinate transformations were obtained as $v = H * u$,

where, $v = [\Theta_1^i, \Theta_2^i, \Theta_3^i, 1]^T$, $u = [X_i, Y_i, Z_i, 1]^T$ and

$H = \mathrm{Trans}(\Theta_1, \Theta_2, \Theta_3)\ \mathrm{Rot}(\Theta_3, 90 + \alpha)\ \mathrm{Rot}(\Theta_1, 90 - \beta)\ \mathrm{Trans}(0,0,Z_i)$

$$H=\begin{bmatrix}1&0&0&\theta_1^s\\0&1&0&\theta_2^s\\0&0&1&\theta_3^s\\0&0&0&1\end{bmatrix}\bullet\begin{bmatrix}\cos(90+\alpha)&-\sin(90+\alpha)&0&0\\\sin(90+\alpha)&\cos(90+\alpha)&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\bullet\begin{bmatrix}1&0&0&0\\0&\cos(90-\beta)&-\sin(90-\beta)&0\\0&\sin(90-\beta)&\cos(90-\beta)&0\\0&0&0&1\end{bmatrix}\bullet\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&Z_i\\0&0&0&1\end{bmatrix}$$

$$H=\begin{bmatrix}-\sin\alpha&\sin\beta\cos\alpha&\cos\beta\cos\alpha&\theta_1^s+Z_i\cos\beta\sin\alpha\\\cos\alpha&-\sin\beta\sin\alpha&\cos\beta\sin\alpha&\theta_2^s+Z_i\cos\beta\sin\alpha\\0&\cos\beta&\sin\beta&\theta_3^s+Z_i\sin\beta\\0&0&0&1\end{bmatrix}$$

$$\bar{v}=H\bullet\bar{u}=\begin{bmatrix}\theta_1^s+Z_i\cos\beta\cos\alpha-X_i\sin\alpha-Y_i\sin\beta\cos\alpha\\\theta_2^s+Z_i\cos\beta\sin\alpha+X_i\cos\alpha-Y_i\sin\beta\sin\alpha\\\theta_3^s+Z_i\sin\beta+Y_i\cos\beta\\1\end{bmatrix}$$

therefore,

$$\Theta_1^i = \Theta_1^s + Z_i * \cos\beta * \cos\alpha - X_i * \sin\alpha - Y_i * \sin\beta * \cos\alpha$$

$$\Theta_2^i = \Theta_2^s + Z_i * \cos\beta * \sin\alpha + X_i * \cos\alpha - Y_i * \sin\beta * \cos\alpha$$

$$\Theta_3^i = \Theta_3^s + Z_i * \sin\beta + Y_i * \cos\beta$$

$$\tan\alpha = \frac{\theta_2^G - \theta_2^S}{\theta_1^G - \theta_1^S}$$

$$\tan\beta = \frac{\theta_3^G - \theta_3^S}{\sqrt{\left(\theta_1^G - \theta_1^S\right)^2 + \left(\theta_2^G - \theta_2^S\right)^2}}$$

Where α and β are defined as in Figure 26, X_i, Y_i are a node coordinate position generated by the GA at plane i, Z_i is the position of plane i along the Start-Goal line.

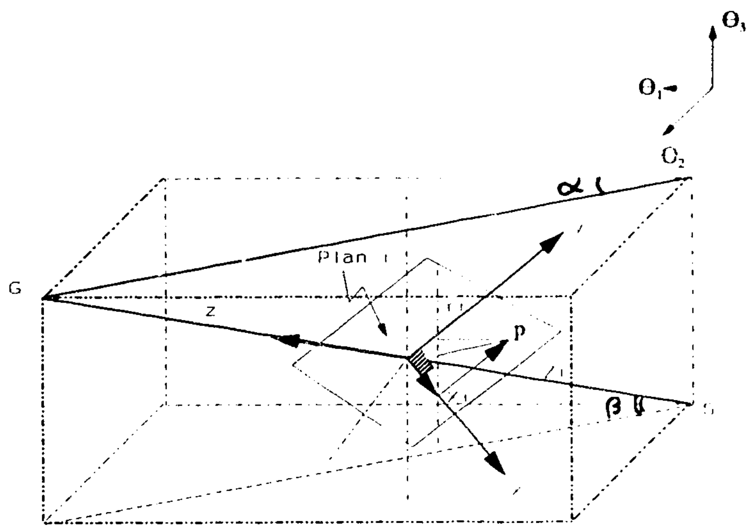


Figure 26. World Space to Joint Space Coordinate Transformation

(7) Collision checking. Every knot point in the generated trajectory is checked for collisions and joint angle limits. The intermediate points along the line between the knot points are also checked for collisions at the same time at an interval set by the user.

(8) Performance evaluation. Fitness function values are evaluated according to the collision checking, joint angle limit checking, and joint displacement minimization as described below,

(i) Minimize the joint angle movement: the joint angle displacement was calculated by

$$\text{Error}_1 = \sum\{(\Theta_{1,i}-\Theta_{1,i-1})^2 + (\Theta_{2,i}-\Theta_{2,i-1})^2 + (\Theta_{3,i}-\Theta_{3,i-1})^2\}$$

i = 0 signifies Start Point

i = N+1 signifies Goal Point

Where $\Theta_{1,i}$,$\Theta_{2,i}$ and $\Theta_{3,i}$ are the three joint angles on the plane i, and $\Theta_{1,i-1}$, $\Theta_{2,i-1}$ and $\Theta_{3,i-1}$ are the three joint angles on the plane i-1.

(ii) Obstacle avoidance: collisions were checked at each node point along a trajectory, and the robot joint angle configurations between any two consecutive knot points were also checked at specified intervals. If there is a collision, a collision counter $\text{No}_{\text{collision}}$ is incremented. The collision counter represents a contribution to the error associated with each trajectory:

$$\text{Error}_2 = \text{No}_{\text{collision}} * \text{Error}_1 \qquad \text{If collision occurs}$$

$$\text{Error}_2 = 0 \qquad \text{No collisions}$$

(iii) Joint angle feasibility checking: the joint angle configuration at each node point will be checked against the allowed joint limits. Any joint angle which is out of the limit will be penalized by increasing the limit counter No_{limit} and adding the No_{limit} to the fitness function value,

$$Error_3 = No_{limit} * Error_1 \qquad \text{Out of the limits}$$

$$Error_3 = 0 \qquad \text{Otherwise}$$

(iiii) The total Error will be

$$Error = K_1 * Error_1 + K_2 * Error_2 + K_3 * Error_3$$

Where, the K's are weighting factors.

The fitness of a trajectory is a relative measure which depends on the performance of other members of the population. The fitness is a positive monotone transformation of error designed to give an appreciative relative reward to different members of the population. An exponential scaling is used so that,

$$Fitness = Exp^{(-Error/k)}$$

where K is a constant used to adjust the fitness values in a reasonable range.

(9) Evolution processing by applying GA operators.

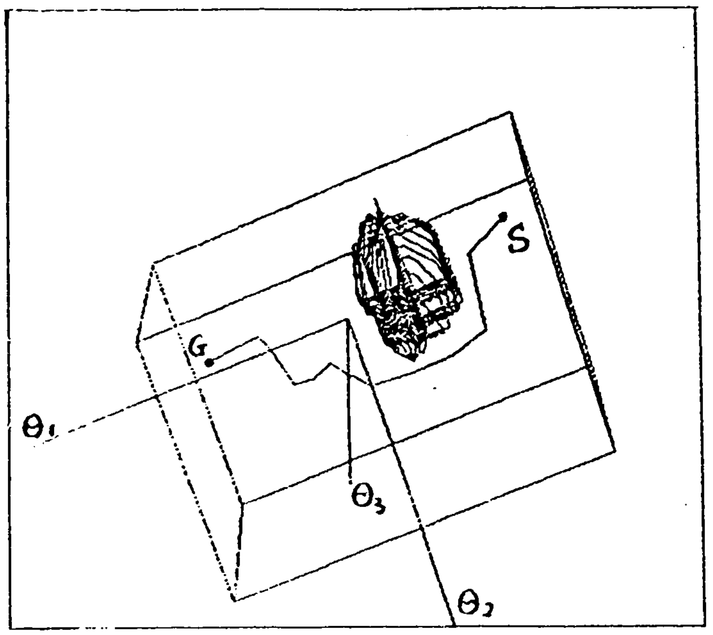
(10) Termination criteria. If the stop conditions are met, the best path in the final generation is recognized as the solution. Otherwise, the evolution process continues.

The simulation was performed to show the effectiveness of the proposed approach for real world application. In the simulation, two workspaces were simulated with two (Figure 27) and five (Figure 29) obstacles. The GA-PATH used 20 strings for the population, and calculated 100 generations with 0.4 probability uniform crossover and 0.002 probability of mutation. The weighting factors were set as $K_1=K_2=K_3=2$ and K=1000. Figure 28 shows the generated path in C-space for the two obstacle workspace, and Figure 30 shows the workspace with five obstacles. The relationship among the maximum, minimum and average fitness of the strings and generations are shown in Figure 31. The GA-PATH demonstrated the successful global path planning ability in very crowded environments, and the robot succeeded in finding the feasible paths with the flavour of global path planning.

4.4 Comparison of GA-PATH Algorithm With PFINDER Algorithms for Mitsubishi RM101 Robot

A comparison was made with the PFINDER path planning algorithm [Toogood and Wong, 1993] which also was applied to the Mitsubishi RM101 robot. The basic operation of the PFINDER is to check at short intervals (1 or 2 degrees) along the straight path between two points in joint space. If a collision is detected, the program first searches around the contour of the C-obstacles on the same Θ_3 plane as the last non-collision point. The search continues until either a path is found around the obstacle, the search is trapped or the joint limits are reached. If the search is trapped or joint limits are reached, the value of Θ_3 is increased (or decreased) and the search is repeated on an adjacent plane. PFINDER was a local path planner based on the heuristic search rules. The GA-PATH, a global path planner, outperformed the PFINDER algorithm in complex space cases. For the same specified start and goal configurations in the clustered environments, PFINDER often became trapped at some local positions (an example is shown in Figure 32) while GA-PATH executed very well as a true global path planner (Figure 29, 30). In some relatively simple workspaces, PFINDER was much faster than

GA-PATH. Note that the time needed for the GA-PATH execution is largely dependent on the GA parameters(such as population and generation size) and string bits numbers, and the GA-PATH algorithm was not coded or optimized for speed. In fact, a very slow language QuickBASIC was used for program development, debugging, and graphics, which is the same as PFINDER.



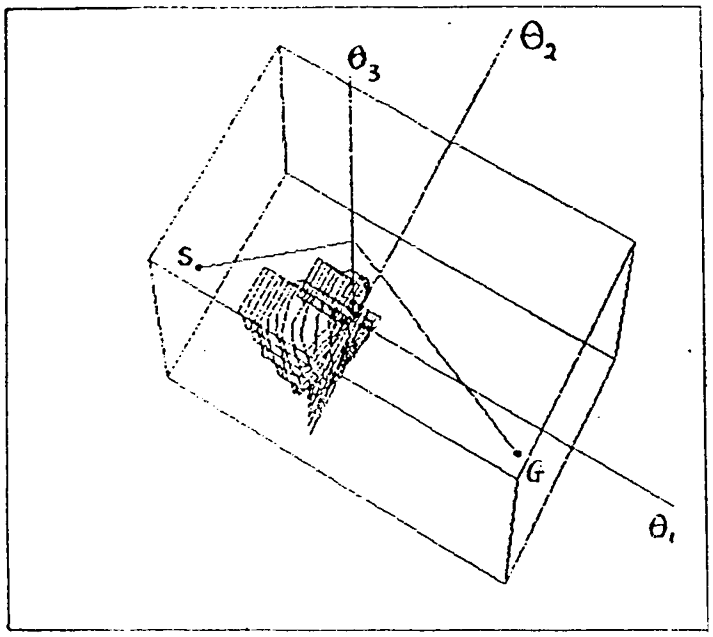


Figure 27. GA-PATH For 2 Obstacles Workspace (Joint Space)

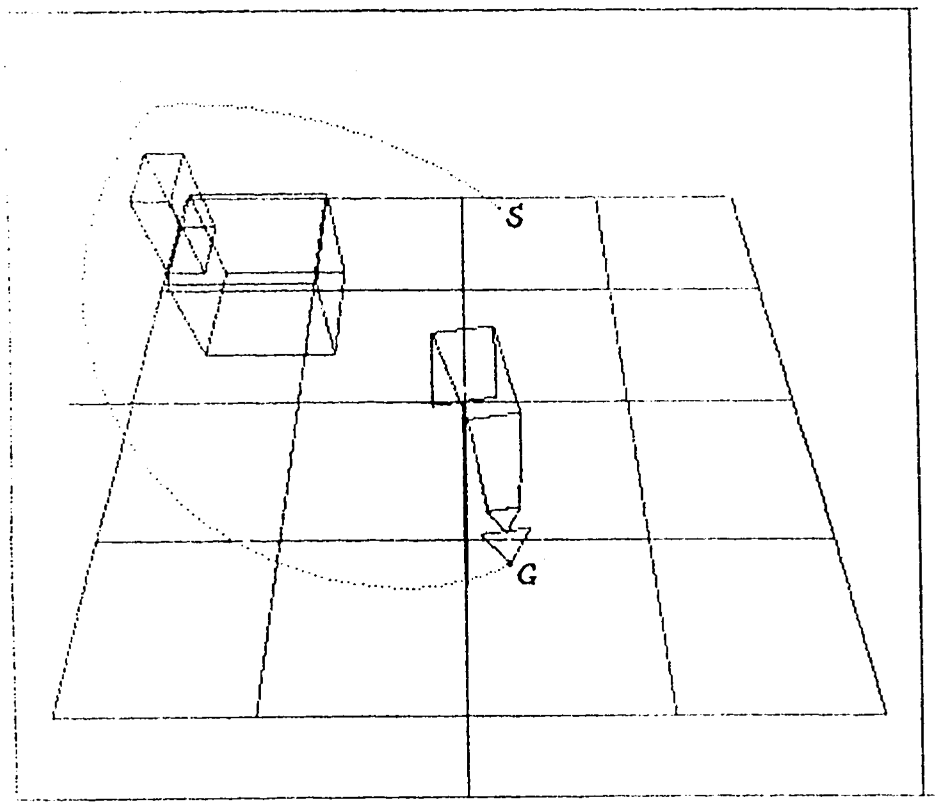
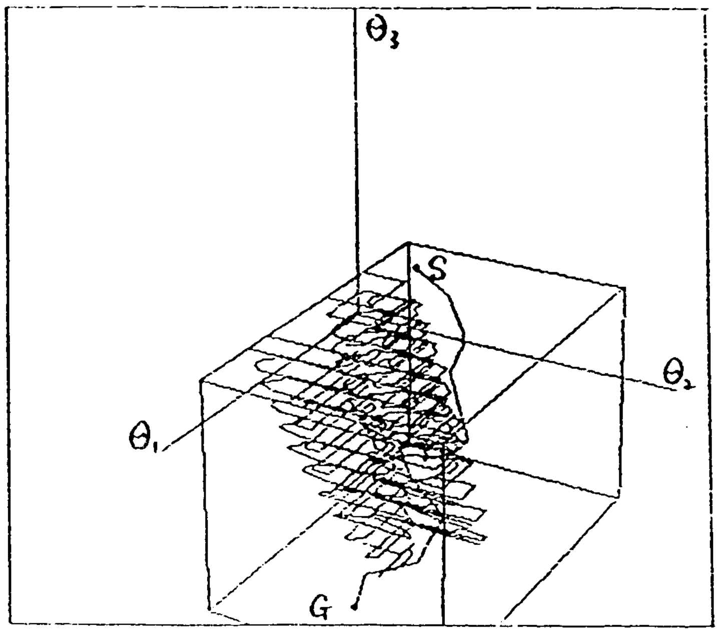


Figure 28. GA-PATH For 2 Obstacles Workspace (World Space)



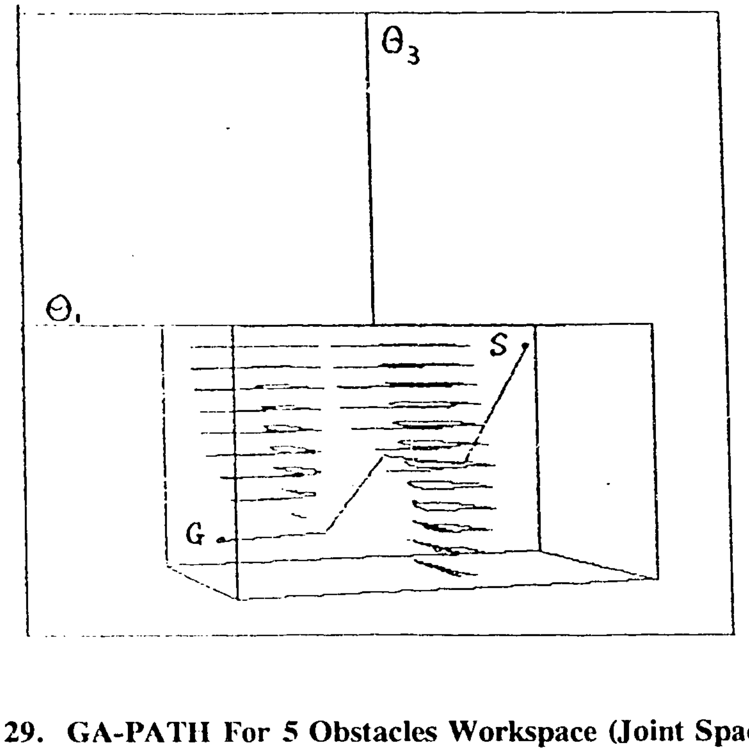
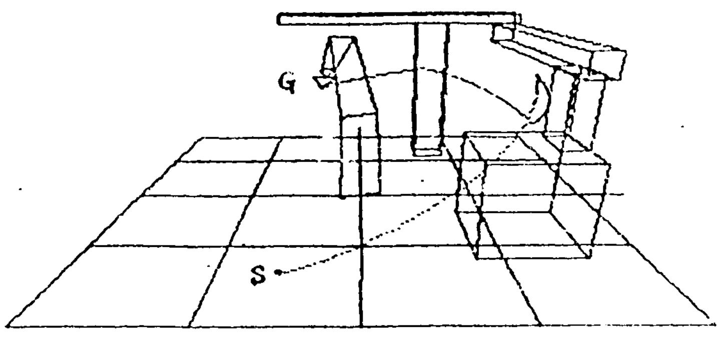


Figure 29. GA-PATH For 5 Obstacles Workspace (Joint Space)



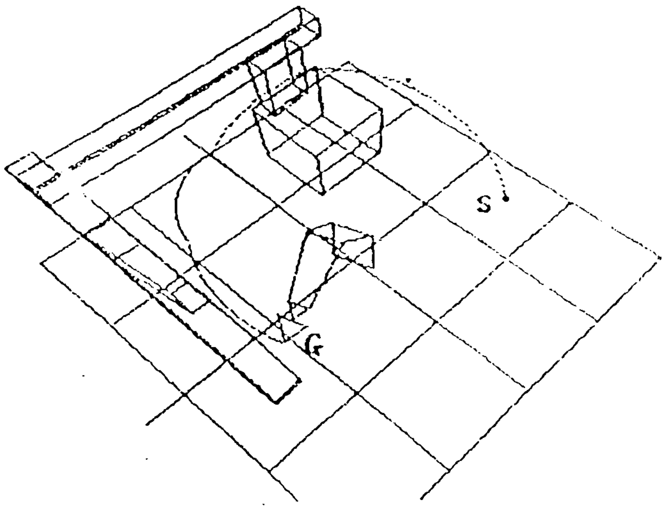


Figure 30. GA-PATH For 5 Obstacles Workspace (World Space)

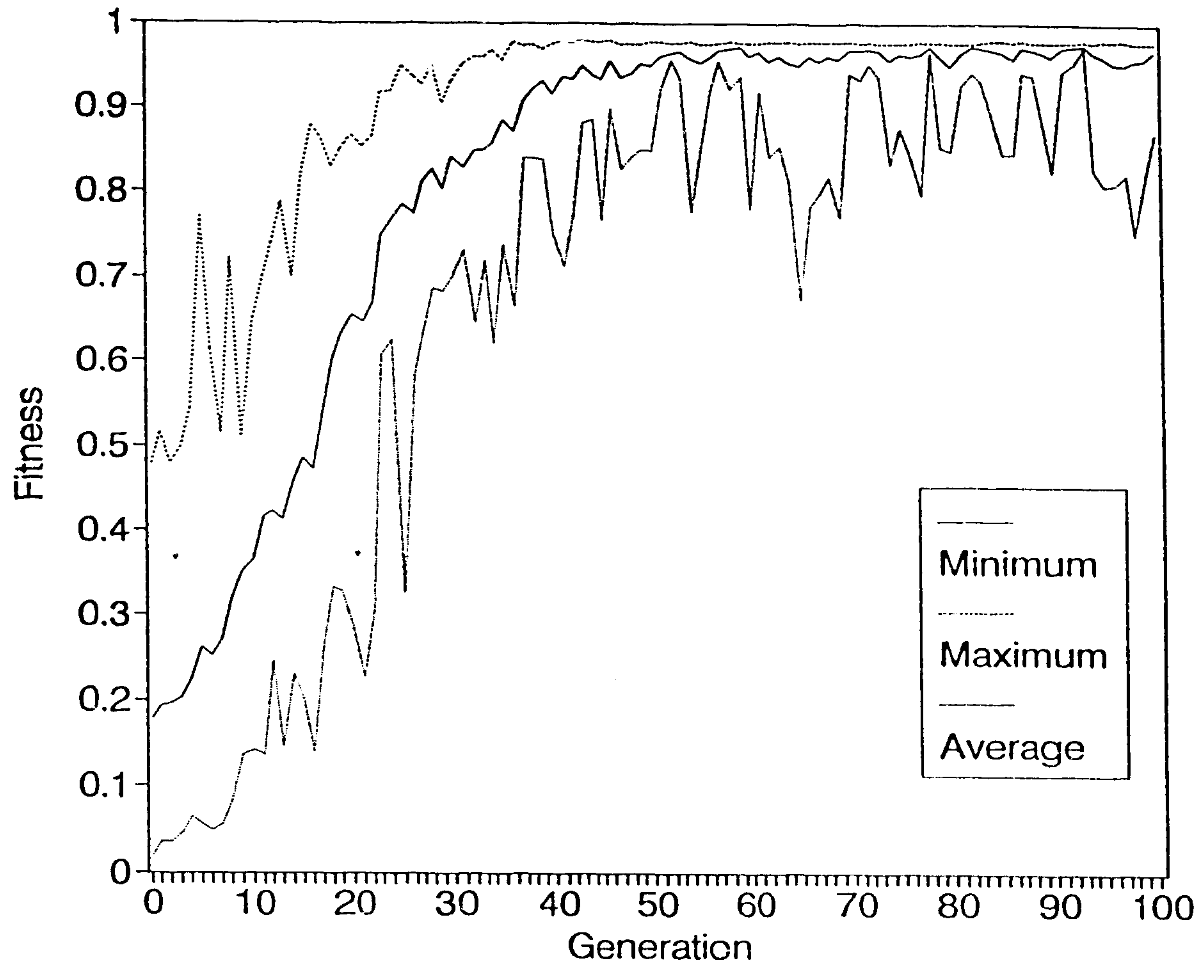
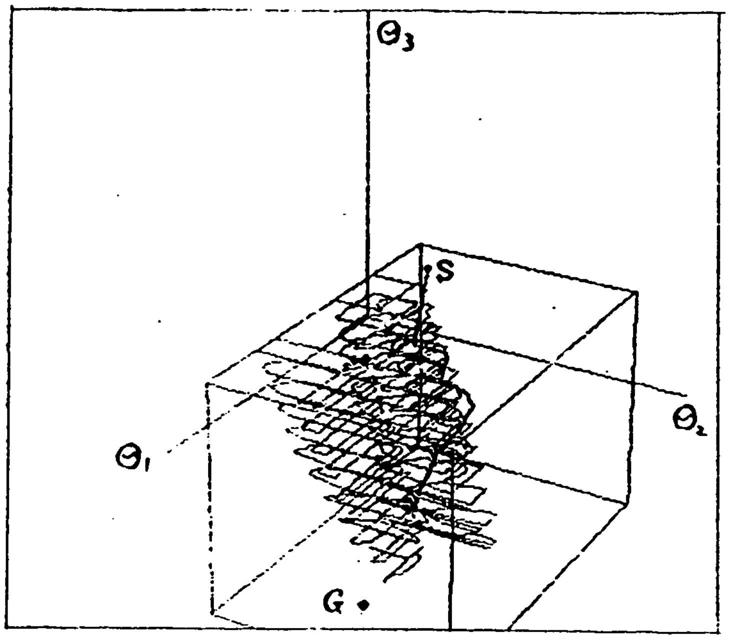


Figure 31. GA-PATH Statistics for Mitsubishi RM101 (n=20, P_c=0.4)



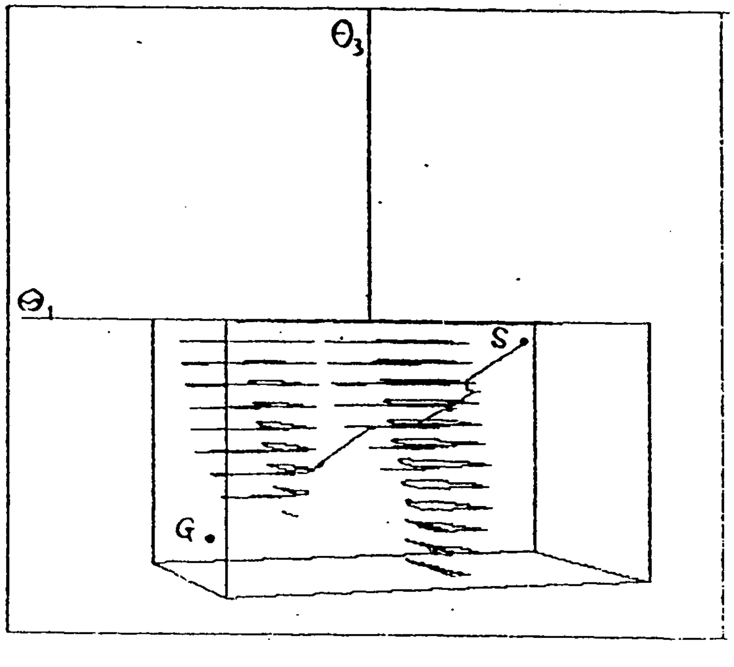


Figure 32. PFINDER For 5 Obstacles Workspace (Joint Space)

5. CONCLUSIONS

Genetic algorithms, which are robust general-purpose optimization techniques, have been used to generate collision-free trajectories for the Mitsubishi RM101 robot with user specified start and goal joint configurations. By minimizing the joint angle displacement for energy conservation, the GA-PATH was able to find feasible (i.e. collision-free) paths globally. The GA-PATH search was performed in C-space while obstacle collisions were computed in world space.

The emphasis of this research was to provide an insight to the basic motion planning problem of generating a collision-free path from one arbitrary point to another by using GAs. The designed GA-PATH algorithm did prove the hypotheses with very promising results.

The key features of GA-PATH are the coding scheme and fitness evaluation. The coding scheme guarantees the end-effector reaches at goal point (unlike Khoogar and Parker's algorithm); the joint angles at knot points $(\Theta_1,\Theta_2,\Theta_3)_i$ specified by coding only two variables $(X, Y)_i$. The fitness evaluation mechanism weighted sum of distance and collision penalties and collisions were determined throughout the path.

Cases with up to five obstacles have been examined within the work space of the robot arm. The limited size of the Mitsubishi's work environment restricted the number of allowable obstacles without overcrowding. Due to the global nature of the GA-PATH algorithm, the path planner successfully found the feasible paths for even a very crowded workspace (5 obstacles).

The uniform crossover operator proved to be capable of handling order dependant representations, and showed its great exploration and exploitation power for searching complex spaces.

The simulation results demonstrated that a GA-based path planning technique has great potential to become a useful searching approach for robot arm/manipulator trajectory generation. The features and the advantages of GAs such as global optimization, good robustness, simple mechanics, large group searching, multi-objective driven as well as intrinsic similarity to natural world will make the GA-based optimization technique an effective and efficient optimization technique in robotics.

Further extensions to this work may include developing a true energy conservation planner which could model the dynamic properties of the Mitsubishi RM101 robot (with mass, inertia, torque or time). Performing a dynamic analysis for a trajectory would allow a computation of the energy expenditure involved in moving from the start to the goal point. In order to separate each individual link's contribution to the fitness function, the following equation could be used simply by adjusting K_{pi}'s.

$$\text{Fitness} = \sum \{ K_{p1} * (\Theta_{1,i}-\Theta_{1,i-1})^2 + K_{p2} * (\Theta_{2,i}-\Theta_{2,i-1})^2 + K_{p3} * (\Theta_{3,i}-\Theta_{3,i-1})^2 \}$$

In the case that the start point is in a trapped situation, an alternative is to add some node points in the opposite direction from the goal, and re-search the feasible path again.

The drawback of GA-PATH as any other GA applications is that it can not absolutely guarantee to find the global optimal path, and the execution time may not meet the needs of real time control requirements. But, the discrete nature of the algorithms used here makes them suitable for implementation on parallel processors (eg. assign one processor per node), which could reduce the computation time drastically.

The code for GA-PATH is available from DR. R. W. Toogood in Department of Mechanical Engineering, University of Alberta, Edmonton, Canada, T6G 2G8.

REFERENCES

1. Beasly, D., D.R. Bull and R.R. Martin. 1993. " An Overview of Genetic Algorithms". University Computing, Vol. 15, No.2.
2. Brady, M. 1982. Robot Motion: Planning and Control. MIT Press, Cambridge, MA.
3. Brooks, R. A., 1983. "Solving the Find-Path Problem by Good Representation of Free Space". IEEE Trans. on Syst. Man Cybern. Vol. 13, No. 3.
4. Cavivhio. 1970. Adaptive Search Using Simulated Evolution. PhD Dissertation, University of Michigan, Ann Arbor.
5. Crow, J. F. and M. Kimura. 1970. An Introduction to Population Genetics Theory. Harper and Row, New York.
6. Davidor, Y. 1991. Genetic Algorithms and Robotics. World Scientific Publishing Co. Pte Ltd., Farrer Road, Singapore.
7. Davis, L. 1991. Handbook of Genetic Algorithms. Van Norstrand Reinhold.
8. DeJong, K. A. 1975. The Analysis and Behaviour of a Class of Genetic Adaptive Systems. PhD dissertation, University of Michigan, Ann Arbor.
9. Grefenstette, J.J. 1986. "Optimization of Control Parameters for Genetic Algorithms", IEEE Trans. Syst., Man, Cyber., Vol. SMC-16, No 1.
10. Goldberg, D. E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Mass.

11. Holland, J.H. 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor.

12. Khoogar, A. R. and J. K. Parker. 1991. "Obstacle Avoidance of Redundant Manipulators Using Genetic Algorithms". Proc. IEEE International Conference on Robotics and Automation.

13. Kwok, D. P., T. P. Leung and F. Sheng. 1993. "Genetic Algorithms for Optimal Dynamic Control of Robot Arms". IEEE IECON Proceedings, V1.

14. Latombe, J.C. 1991. Robot Motion Planning. Kluwer Academic Publishers, Norwell, Massuchusetts.

15. Lozano-Perez, T. and M. A. Wesley. 1979. "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles". Communications of the ACM, Vol.22, No. 10.

16. Parker, J. K. and D. E. Goldberg. 1989. "Inverse Kinematics of Redundant Robots Using Genetic Algorithms". Proc. IEEE International Conference on Robotics and Automation.

17. Parkin, R.E. 1991. Applied Robotic Analysis. Prentice Hall, Englewood Cliffs, NJ.

18. Ram, A., R. Arkin, G. Boone and M. Pearce. 1994. "Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation". Adaptive Behaviour, vol. 2, No. 3.

19. Schwartz, J. T. and M. Sharir. 1988. "A Survey of Motion Planning and Related Geometric Algorithms". Artificial Intelligence(37).

20. Shibata, T. and T. Fukuda. 1993. "Coordinative Behaviour by Genetic Algorithm and Fuzzy in Evolutionary Multi-Agent System". IEEE International Conference on Robotics and Automation.

21. Spears, W. and K. A. DeJong. 1991. "On the Virtues of Parameterized Uniform Crossover". Proc. 4th. International Conference on Genetic Algorithms.

22. Syswerda, G. 1989. "Uniform Crossover in Genetic Algorithms". Proc. 3rd Int'l Conference on Genetic Algorithms.

23. Takahashi, O. and R. J. Schilling. 1989. "Motion Planning in a Plane Using Generalized Voronoi Diagrams". IEEE Trans. on Robotics and Automation, Vol.5, No. 2, April.

24. Toogood, R.W. 1991. "A Work Cell Animator for Robotics Instruction", ASME Computers in Engineering, Santa Clara, CA, V-2.

25. Toogood, R.W. and C. Wong. 1993. "Robot Obstacle Avoidance and Path Planning in Configuration Space", ASME Computers in Engineering Conference, Santa Barbara.

26. Unimation Inc. 1987. Unimate PUMA MARK III Robot Models 552/562 Equipment Manual #398AH1. Danbury, CT.

27. Wong, C. 1992. Robot Path Planning In Configuration Space. M.Eng. thesis, University of Alberta, Edmonton, Canada.

28. Zhao, M., N. Ansari and E. S. H. Hou. 1994. "Mobile Manipulator Path Planning by A Genetic Algorithm". Journal of Robotic Systems 11(3).