Multi-objective Optimal Path Planning Using Elitist Non-dominated Sorting Genetic Algorithms

Faez Ahmed and Kalyanmoy Deb[*]
Kanpur Genetic Algorithms Laboratory (KanGAL)
Department of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur, PIN 208016, India
{faez,deb}@iitk.ac.in
http://www.iitk.ac.in/kangal/pub.htm

KanGAL Report Number 2011013

Abstract

A multi-objective vehicle path planning method has been proposed to optimize path length, path safety and path smoothness using the elitist non-dominated sorting genetic algorithm (NSGA-II). Four different path representation schemes that begin its coding from the start point and move one grid at a time towards the destination point are proposed. Minimization of traveled distance, maximization of path safety, and maximization of path smoothness are considered as objectives of this study. A performance evaluation measure using the hypervolume metric of the obtained trade-off front is used. This study makes a careful analysis of a number of issues related to the optimization of path planning task – handling of constraints associated with the problem, handling single versus multiple objectives, performing a parametric study to locate reasonable values of key parameters, identifying an efficient path representation scheme, and evaluating the proposed algorithm on large sized grids and having a dense set of obstacles. The study also compares the performance of the proposed algorithm with an existing GA-based approach. The evaluation of the proposed procedure against extreme conditions having a dense (as high as 91%) placement of obstacles indicates its robustness and efficiency in solving complex path planning problems.

1 Introduction

In recent years we have seen much advancement in the field of industrial robotics, but mobile intelligent machines have mainly been confined to research labs. For an autonomous vehicle to be used in real-world applications it must be able to navigate autonomously and take intelligent decisions. Vehicle path planning comprises of not only generating collision-free paths from a given location to its destination point but also finding an optimal path that minimizes or maximizes certain critical objectives. References [25] and [22] provide a broad coverage of the field of motion planning algorithms focusing on vehicle motion planning.

There have been many conventional methods for two-dimensional path planning using classical optimization methods, [3, 35], artificial potential field method [18, 2], visibility graph [26, 27], Voronoi road-map [6] etc. Many artificial intelligence techniques like neural networks [20, 36], fuzzy logic method [28, 29], genetic algorithms [19, 4] and ant colony optimization [30] have

[*]Also Adjunct Professor, Department of Information and Service Economy, Aalto University School of Economics. 00100 Helsinki, Finland

recently come to forefront for path planning. In a path planning task, a majority of research has been dedicated to finding a feasible shortest path by formulating a single-objective optimization problem. However, finding the optimal path is never a single-objective problem, as many other attributes such as path safety (or path vulnerability) and path smoothness are also desirable for a navigation vehicle. The path vulnerability objective is associated with how close the vehicle pass through an obstacle. To compute this objective, we have taken help of the artificial potential field concept. Each obstacle holds a large potential in its center and the potential is assumed to reduce away from the obstacle. Thus, for a given path and a given setting of obstacles, the overall potential computed for the entire path accumulated due to the effect of all obstacles is denoted as the path vulnerability. Path smoothness is directly related to the amount of turn a vehicle has to take along the path from the start to the end of its motion.

Besides the single versus multiple conflicting objectives associated with the path planning problem, there are a number of other issues that must be resolved during an optimization process. One important matter is the scheme used to represent a path within the optimization procedure, as the performance of an optimization algorithm depends largely on the chosen representation scheme. In this paper, we propose four different techniques for path representation based on a binary, mixed or integer coding of a path description. Second, the issue of a two-objective versus a three-objective optimization is another matter that will concern the practitioner and will test the usefulness of the chosen algorithm. Third, the use of obstacle avoidance as a hard constraint in the optimization process or handling of it softly through a penalty term is another matter that will determine the performance of the chosen algorithm. Fourth, every optimization algorithm introduces some parameters which often controls the outcome of a simulation run. Thus, a parametric study of key parameters is essential to understand the effect of parameters on the performance of the algorithm so that a suitable setting of the parameters can be used for a real application. Finally, for a path planning problem, the scaling of the chosen optimization algorithm with the number of obstacles and the grid size must be well understood before the method can be recommended for a practical use. In this paper, we address all the above issues related to the path planning problem one at a time by considering obstacle scenarios that were adopted in earlier path planning studies. A bi-objective algorithm minimizing path length and path vulnerability is proposed based on the elitist non-dominated sorting GA (NSGA-II) [12], but the algorithm is modified to use the third objective (path smoothness) as a decision making aid for identifying less-crowded solutions. To provide a stiff challenge to the proposed algorithm, the complexities of obstacle placement is gradually increased to have as high as 91% of the cells occupied by obstacles. For this purpose, the search domain is also increased from 8×8 to a staggering 128×128. These extremities are rarely used in the past computational path planning studies and the successful working of the proposed algorithm in them amply demonstrates their practical importance.

In the remainder of the paper, four new path representation techniques are discussed in Section 2. Three objectives used in this study are described next in Section 3. The chosen single and multi-objective genetic algorithm approaches including a couple of constraint handling methodologies are discussed in Section 4. Thereafter, Section 5 performs a parametric study on a previously attempted problem to identify reasonable values of key parameters associated with the proposed algorithm. Section 6 compares different representation schemes suggested in this paper. Section 7 makes random placements of obstacles on differently sized grids and investigates the efficiency of the proposed algorithm in finding a feasible path in the presence of an increasing number of obstacles. Problems having as large as 128×128 grids and 91% coverage of grids with obstacles, thereby resembling extreme practical scenarios, are considered next. Conclusions of the study are made in Section 8.

2 Path Representation Through a Grid

To optimize any given path of a vehicle through an optimization technique, the foremost requirement is an efficient path representation that can be incorporated in the programming environment and importantly that is suitable to be efficiently used with the chosen optimization algorithm. In this study, we use a genetic algorithm (GA) as an optimization algorithm. Since a GA allows a flexible way to represent a solution, we suggest four different representation schemes for the purpose.

To facilitate a discrete representation and also for the ease of navigation purposes, a grid based approach is adopted here. In a given problem of 2-D off-line path planning, we map the environment on a grid of size $n \times n$, where the grid size n is decided depending on the accuracy required in the path planning problem. The size of the vehicle is taken to be smaller than unit cell size in grid so that the dynamics of vehicle rotation is not to be considered. For simplicity, we always take the start location as the bottom-left corner, denoting grid location $(1, 1)$. In the Cartesian coordinate system, the destination is then fixed as (n, n) (or the top-right corner of the grid). This assumption is a reasonable one since most other path planning scenario can be mapped on to such a grid by simple linear transformations, such as translation, rotation, and scaling. Now if we consider the characteristics of the entire path between a start location and final location then different paths taken by a vehicle may have different lengths, directions, and turns at different points. A possible solution is a linked-list representation [34] of the entire path having a variable length gene. Such a representation of a path by genes of variable length is cumbersome considering various GA operations, like crossover as well as the absence of knowledge about the bounds of path length. A generalized representation would considerably increase the search space hence slowing down the search process. Considering these difficulties with variable-length gene representations, we propose a fixed-length path representation here. Motivated by another study [33], we consider only paths which are monotonous along one axis. That is, starting from location (1,1), the path moves one grid at a time towards right till it reaches the destination point (n, n). Since there are n columns in the search domain and the vehicle is already at the first column, we would require $(n - 1)$ genes to represent a path. We propose three different techniques of representing a path involving binary, integer, or mixed-valued genes. We describe these representation schemes in the following subsections.

2.1 Binary-Coded Genes

Our proposed path representation technique was inspired by a recent study [33]. We first describe the procedure and then shall describe our coding method to alleviate a difficulty associated with the earlier procedure. Let us consider the grid as comprised of rows and columns. To represent a path, following symbols were defined:

- 'A' denotes an upper diagonal step from one column to the next parallel to $(\hat{\imath} + \hat{\jmath})$.
- 'B' denotes a lower diagonal step from one column to the next parallel to $(\hat{\imath} - \hat{\jmath})$.
- 'C' denotes a horizontal step from one column to the next parallel to $(\hat{\imath})$.
- 'Dy' denotes an upward vertical movement by y units within a column parallel to $(+\hat{\jmath})$.
- 'Ey' denotes downward vertical movement by y units within a column parallel to $(-\hat{\jmath})$.

Graphical notion of the above symbols is given in Figure 1, where $\hat{\imath}$ corresponds to positive x-axis and $\hat{\jmath}$ to positive y-axis. A gene comprises of $(n - 1)$ movements where each movement is one transition step (A, B or C) followed by a vertical motion (Dy or Ey) within a column, where y denotes magnitude of vertical motion. In [33], transition from one column to other was either through A, B or C, but any vertical movement Dy or Ey within a column was ignored if the last transition was either A or B. That is, if a diagonal move is taken, the path cannot go upwards or downwards, it can only go forward, whereas to move up more than one grid up or down, the only option is to go horizontally towards right and then make a vertical move up or down. This is restrictive and as we shall discuss later it will create paths that are dominated by diagonal paths. We modify the above coding by allowing vertical steps after type A or B movements as well.

In our proposed binary-coding, each gene comprises of two parts. First two bits represent the direction of transition to the next column in four possible ways. The vehicle can move diagonally up by one grid and then a distance y up along the column. We call this movement A-Dy and is represented by a `11` substring followed by a second substring decoding to y. Since n is the maximum grid size, $\lceil \log_2 n \rceil$ bits are needed to represent y. Thus, a movement by one grid towards right requires $(2 + \lceil \log_2 n \rceil)$ bits. This and other three scenarios are depicted in Figure 1.

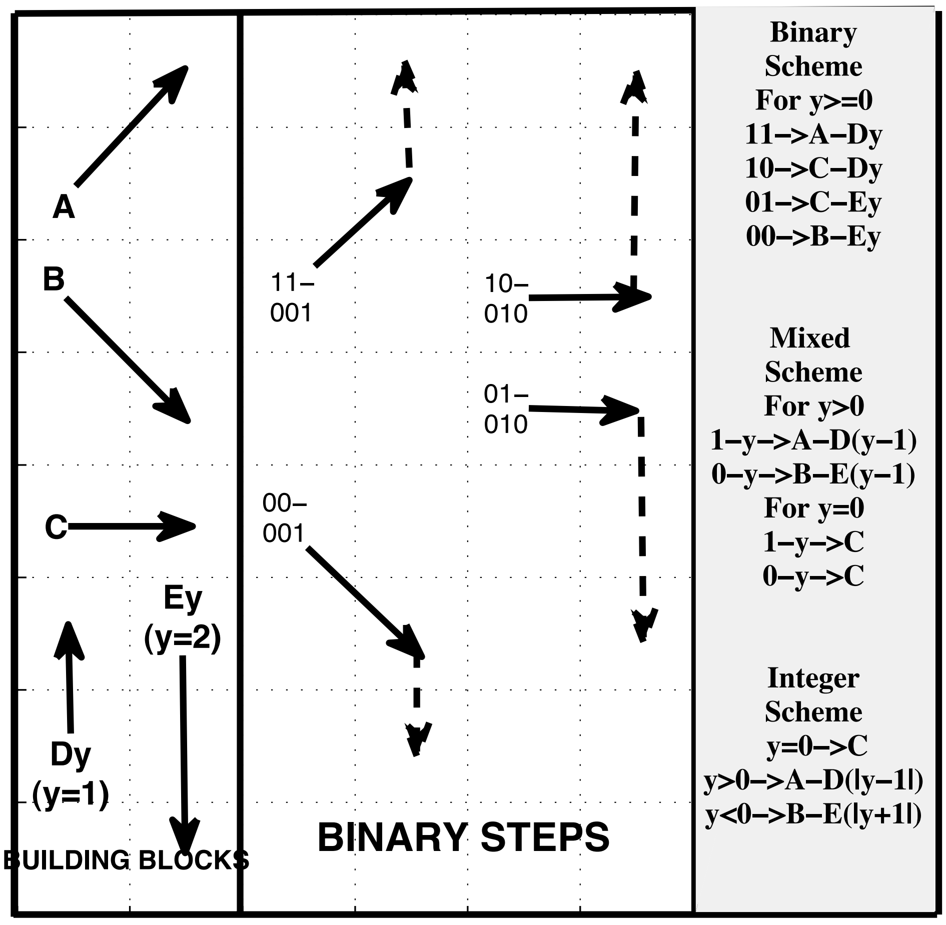


Figure 1: Representation scheme denoting (Left) Steps corresponding to alphabetic codewords, (Center) Decoded binary bits, and (Right) Representation schemes of transition followed by vertical movement step in terms of alphabetic codewords.

Similarly, B-Ey would represent a move diagonally down by one grid and then a further movement down by y grids. This process is represented by a (`00` followed by $\lceil \log_2 n \rceil$) bits decoding to y. Substrings (`10` or (`01` represents a horizontal movement by one grid, but each is integrated with a further upward or a downward movement of y grids. The upward movement is denoted as C-Dy and downward movement is by C-Ey. Thus, to represent all $(n-1)$ grid movements towards right from start to destination, a total of $(n-1)(2+\lceil\log_2 n\rceil)$ bits would be required.

A careful thought will reveal that by the above representation scheme, we have eliminated a diagonally up movement followed by a downward movement or A-Ey, as the same can be achieved with a smaller path length and path safety using a C-Dy or a C-Ey movement. Since in our multi-objective formulation, A-Ey would always be dominated by either C-Dy or C-Ey, we simply do not allow A-Ey or B-Dy movements. Thus, this representation scheme reduces the feasible search space and in turn helps the optimization algorithm to work better.

Next, we discuss an end-fixing mechanism that goes with the above representation scheme. It is mentioned above that every path originates from the start point $(1,1)$, but following a particular coding of $(n-1)$ genes (as mentioned above) every path may not end up reaching the destination point (n,n). To make sure that every path reaches the destination point, we modify the $(n-1)$-th gene as follows. The path is followed by the first substring of the $(n-1)$-th gene, but the value of y from the second substring is ignored. A (`11`) will take the path one grid diagonally up, a (`00`) will take the path one grid diagonally down, and (`10`) or (`01`) will take the path horizontally by one grid. Thereafter, instead of using y to determine whether to go up or down, the path is simply joined to the destination point. Since the second substring coded to represent y for the $(n-1)$-th gene is redundant, we use it to represent any upward movement of the vehicle at the *start* grid. We take a couple of examples to illustrate this representation scheme.

In Figure 2, two sample paths are shown on a 8×8 grid. Thus, $n=8$ in this example. The binary representation of Path 1 is given as follows:

```
(000, 11-010, 11-000, 10-000, 00-001, 00-000, 10-000, 11)
```

Note that the seventh gene is split into two, of which the first substring indicating a diagonally upward movement from seventh to eighth grid is represented by the final (`11`) and its associated substring (`000`) is shown at the beginning to indicate that there is no vertical movement of the path at the start grid. Other movements – for example, from first to second grid there is an upward diagonal movement (decoded by (`11`)) followed by two grids (decoded value of (`010`) of upward movement (A-D2), and others – are straightforward. Notice how the end-fixing on the right-most column takes the path to the destination point. The vertical movement of five grids is not coded in the string, but is fixed algorithmically, making sure that the overall path goes from the start to the end grid. On the same manner, Path 2 can be represented by the following binary string:

```
(101, 11-000, 01-000, 10-000, 00-001, 00-000, 11-001, 11)
```

Here, notice that at the start the vehicle moves by five grids (decoded value of (`101`) shown at the front of the string) at the start and then follows the other grid-wise movements. At the final grid, the vehicle makes an upward diagonal movement and reaches the destination point on its own. Thus, in this path, no end-fixing is needed for the path to reach the destination point.

In addition to the end-fixing involving the start and the n-th columns, if at any intermediate column the vehicle goes out of the search area (to the bottom of first row or to the top of n-th row), it is brought back to the search area by using an edge-fixing procedure. In this event, the y component of the string is ignored and the vehicle is placed randomly at a row with a linear probability distribution having its maximum at the respective edge of the search area to zero at the current row of the vehicle. In both Paths 1 and 2, this scenario did not occur and no edge-fixing was required.

The advantage with the above binary coding is that a binary-coded genetic algorithm can now be used directly to evolve the population members. The usual single or multi-point crossover operators and the bit-wise mutation operator can be applied directly. Here, we have used a

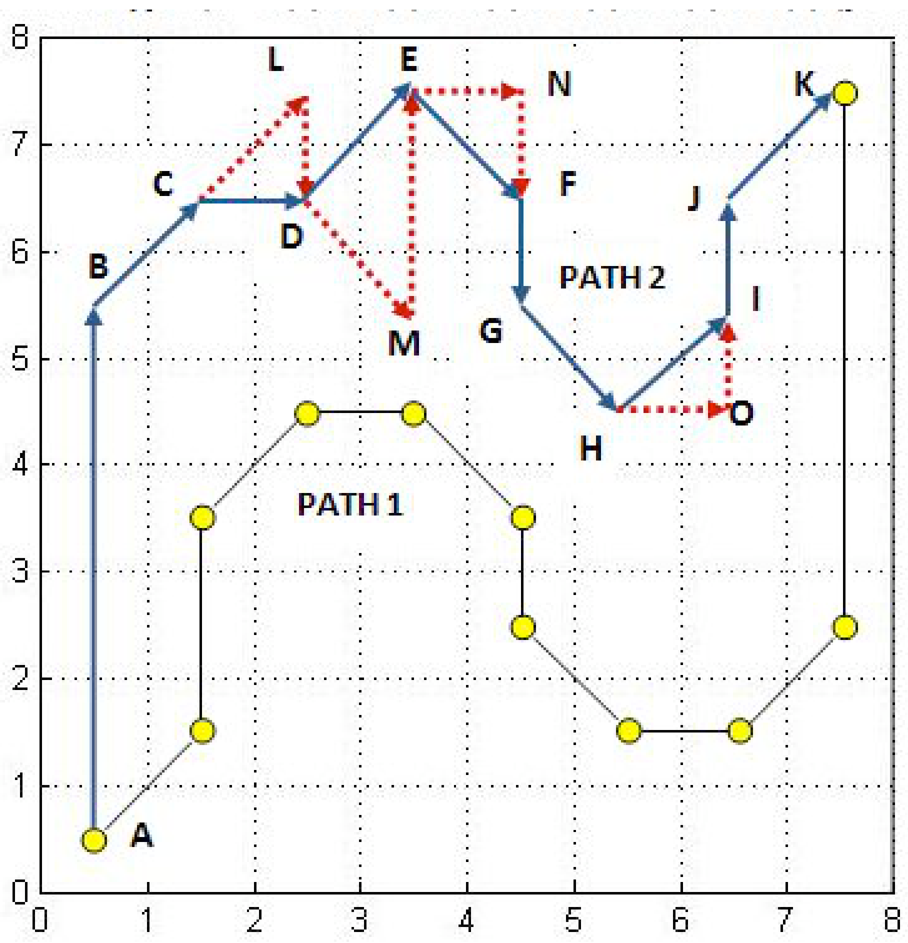


Figure 2: Example paths with corresponding binary representation along with a few non-represented steps in path.

two-point crossover and a bit-wise mutation operator for this purpose, however, the mutation probability is linearly reduced to 1/5-th of its initial value at the final generation. Moreover, since the search space is substantially reduced by not allowing intuitively dominated paths to appear through the representation scheme, the search is also likely to be efficient. The fix-ups mentioned above will also make the paths realizable. The drive towards feasible and eventually optimal paths will then be achieved by the way of handling the objective function and constraints.

2.2 Mixed-Coded Genes

The above binary-coded representation makes a significant improvement from the existing study [33] in terms of reducing the dominated paths. However, some thoughts will reveal that the coding can be made more efficient by using a mixed binary-integer representation of a path.

Consider, for example, a C-D2 path going from (1,1) to (2,1) to (2,3). This path is dominated by another path (A-D1) which makes an upward diagonal movement and then move up vertically by one grid. This path is shorter than C-D2 and since the vehicle is considered to be small compared to the grid dimension, it is also as safe as C-D2. Thus, on both account, A-D1 is no worse than C-D2, and hence C-D2 can be considered redundant in our search process. To eliminate such dominated paths to appear through our coding scheme, here we use one bit to represent upward or downward movement and a direct integer value to represent the extent of vertical movement, if any. We discuss this mixed-coding scheme in the following paragraph.

The mixed-coding representation for a gene has two components as before. But here, the first component has just one bit. However, the meaning of the bit depends on the associated second component – an integer in the range $[0, n]$ (both inclusive) representing vertical movement y. If $y = 0$, a `1` or a `0` in the first substring would mean an identical movement (C), that is, a movement of one grid horizontally to the right. However, if $y > 0$, then a `1` in the first component would mean a move diagonally upwards and then make $(y - 1)$ vertical grids up. Thus, `1`-y means A-D$(y - 1)$ for $y \geq 1$. Similarly, a `0` with a non-zero y would mean a downward diagonal movement followed by $(y - 1)$ vertical grids down. Thus, `0`-y means B-E$(y - 1)$. This mixed-coding does not allow the path to have a horizontal move followed by any vertical upwards or downwards movement, as they were allowed in the binary-coded representation. Any up or downward movement must be achieved with a non-zero y at any gene. This restriction does not prevent any feasible non-dominated sub-path to be represented. This substantially reduce the number of dominated solutions in the search space.

Path 1 shown in Figure 2 would now be represented by the following coding:

(0 `1`-3 `1`-1 `0`-0 `0`-2 `0`-1 `1`-0 `1`)

Like before, the first element (a zero) indicates no vertical movement at the start grid. Next mixed-gene indicates that an upward diagonal movement and then a vertical movement of (3-1) or two grids up. The path can then be followed by the above-mentioned coding principle. At the $(n - 1)$-th column, a `1` indicates a diagonal movement upwards. A `0` would have indicated a horizontal movement to the n-th column. Since the seventh gene is a `1`, the vehicle moves one grid diagonally upwards. From this position, the end-fixing rule is applied and the vehicle is moved up another five grids up to reach the destination point. Path 2 can be represented as follows:

(5 `1`-1 `1`-0 `1`-1 `0`-2 `0`-1 `1`-2 `1`)

The edge-fixing and top-row-fixing rules are applicable here as well. Although the above gene representation makes the explanation of the path easier, in a GA, we concatenate all bits together and form a $(n - 1)$-bit string and all integer values are put together to form a $(n - 1)$-size integer vector. The binary string is operated by the standard single-point crossover and bit-wise mutation operators and the integer-valued vector is operated by the discrete versions of the simulated binary crossover (SBX) [10, 13] and the polynomial mutation operator [9].

2.3 Integer-Coded Genes

It is clear from the above mixed-coded representation that there is a still a small redundancy in the coding. A horizontal movement by one grid can be represented by using either `0`-0 or `1`-0. As mentioned above, as long as $y = 0$, the value of the preceding bit was not important. We remove this redundancy in the mixed-coding scheme by completely eliminating the binary coding and representing y of each gene by an integer in the range $[-(n - 1), (n - 1)]$. Only in the first gene, y values in the range $[0, (n - 1)]$ is allowed. Other end-fixing, edge-fixing and top-row-fixing rules are applicable here as well. Thus, to represent Path 1 in Figure 2, the following $(n - 1)$-variable vector would be used:

$$(0,\ 3,\ 1,\ 0,\ -2,\ -1,\ 0)$$

Here, the first element represents the movement at the start column and the final element represents the movement at the $(n - 2)$-th column to $(n - 1)$-th column. The movement from $(n - 1)$-th to n-th column has just two options before the end-fixing operation. The vehicle can go either upwards and a possible vertical movement thereafter to reach the destination point or a horizontal movement to directly reach the destination point. This is because a horizontal move followed by a vertical move always gets dominated in our set-up by a direct diagonal movement. At the end of string coding, we investigate the string for either of the two scenarios and arrange the path accordingly. For Path 1, we need to move diagonally one grid up, as the current location of the vehicle is below the destination point and then move five grids vertically upwards to reach the destination point. Thus, the transition from $(n-1)$-th column to the n-th column is never coded in the proposed integer-coding scheme and is fixed based on the above argument. Path 2 can be represented by the following integer-coding:

$$(5,\ 1,\ 0,\ 1,\ -2,\ -1,\ 2)$$

This path also requires a diagonal movement from seventh column to the eighth column, as the location of the vehicle at the seventh column is below to that of the destination point.

To demonstrate the difference between the three coding schemes discussed above, let us consider a couple of alternate paths shown in Figure 2. Let us denote ABCDENFGHIJK as Path 3 and ABCDEFGHOIJK as Path 4. These paths can be represented by our binary-coded representation scheme, but cannot be represented by our mixed or integer-coded scheme. In Path 3, the part ENFG takes the vehicle from the initial cell E to cell G in the next column. This can always be replaced by EFG to reach the same location with a shorter path length and path vulnerability (to be defined in a later section). Similarly, in Path 4, the part HI dominates the part HOI. Therefore, Paths 3 and 4 are dominated by another path and can never be on the true Pareto-optimal front.

2.4 Absolutely-positioned Integer-Coded Genes

As a fourth representation scheme, the integer-valued genes described in the previous subsection can be replaced with the absolute location of the grids along each column. Since a path always starts from the grid location (1,1), a set of $(n-1)$ integer values in the range $[1, n]$ will determine the path exactly. The final transition from $(n-1)$-th column to the n-th column can be made by using the procedure described in the previous subsection. A similar representation scheme was suggested elsewhere [17].

The integer-valued coding in the previous subsection is sensitive to the initial part of the path. A small change in the initial part may make a significant change in the final part of the path, as every gene represents a relative change in location from the previous gene. The absolute-positioned integer-coding scheme described here does not have such a difficulty. However, when we perform simulation runs on some simpler scenarios, we noticed the inadequacy of this absolutely-positioned integer-coded representation. Due to the insensitivity of each gene over the other, the relative movement between one column to the next is not important in this coding and this leads to a zig-zag path and the path often results in infeasible paths. Since a genetic algorithm is believed to work by finding lower-order building blocks to form higher-order building blocks, the relative representation may provide a GA to focus on the initial part of the path first and then may enable to form the remaining path grid by grid till the complete path is formed. Such an implicit ordering in forming the optimal path may turn out to be an important property for the successful working of a GA. Discouraged by these proof-of-principle experiments, we have not continued with the absolutely-positioned integer-coded gene representation scheme in this study.

3 Objectives of the Path Planning Problem

In this paper, we have considered three specific objectives of the path planning problem.

3.1 Minimum Path Length

Every path planning problem must provide some importance to a shorter path. In this study, since a path is represented on a rectangular grid, we simply compute the path length by summing the distance between consecutive grid points along the path. The vehicle is always considered to be traversing through the center of a grid cell, thus, a distance between $((i, j)$-th grid to (k, l) grid (where $i, j, k, l \in [1, n]$) is computed as the Euclidean distance between coordinate points $(i - 0.5, j - 0.5)$ to $(k - 0.5, l - 0.5)$.

3.2 Minimum Path Vulnerability

The flip side of following a shortest path is the danger of colliding with an obstacle. To estimate how precariously the vehicle moves close to obstacles, we follow the commonly-used potential field approach. The potential value in a cell represents directly the difficulty in traversing through it. Since each cell is allotted a potential value according to obstacle distribution, we sum the potential of each cell through which the path traverses to compute the total vulnerability level.

The potential field approach is a widely studied method [1, 7, 24]. It gained popularity because it displayed a great reduction in computational demand when compared to the configuration space approach. In this method, each obstacle in the system is surrounded by a repulsive potential field whilst the goal location is surrounded by an attractive potential field. A vehicle applies the resulting force experienced at its location as a control input to the driving system. The approach however, suffers from a major problem. This has to do with the formation of local potential minima that can trap the vehicle before it reaches its destination point. In this study, we have used practical approach of assigning a bell-shaped (Gaussian) potential field at every cell containing an obstacle. A variance of 50% of the cell size is used here, however higher or lower values of variance can be used to have an effect of more or less conservative paths, respectively.

3.3 Path Smoothness

The smoothness of a path is evaluated by summing the angle of each turn the vehicle has to make in traversing the path. The smoothness of a trajectory is a desired attribute in vehicle path planning [23]. Smooth paths decrease unnecessary curvature discontinuity and possible stops, and thus lowers the possibility of slippage. It leads to lesser power consumption as well as lesser travel time. Though smoothness is desired in a path, it need not be used as an objective function in most path planning optimization tasks, as the true optimal smooth path that ignores path vulnerability and other maneuvering issues may not be what a path planner would be looking for.

In this study, we consider smoothness as a secondary goal of the path planning task. In a bi-objective optimization framework of minimizing path length and path vulnerability, path smoothness can be used as a tie-breaker or as a decision-making aid. We shall discuss the interaction of path smoothness with the other two objectives in the next section.

4 Optimization Methodologies

In this section, we discuss various optimization methodologies developed for solving the path planning problems from various perspectives.

4.1 Obstacle Avoidance Procedures

It is intuitive to keep constraint from interfering with an obstacle. In our algorithms, we have used a potential function for this purpose. Each cell having an obstacle has a high potential at its center and it reduces away from the center. Thus, any path that crosses a cell having an obstacle will accumulate a large potential value. Thereafter, using the overall potential (called as path vulnerability above) in the optimization problem either as an objective to be minimized or a constraint to be satisfied, the optimization process will drive the search towards paths that are obstacle-free.

However, in an extended version of our optimization algorithms, in addition, we count the number of obstacles a path crosses and call this number the 'Number of interfering cells'. The multi-objective optimization problem is modified as follows:

$$\begin{array}{rl} \text{Minimize} & \text{Path length}, \\ \text{Minimize} & \text{Path vulnerability} \\ \text{Subject to} & \text{Number of interfering cells} \leq 0, \end{array} \tag{1}$$

and the single-objective problem is modified as follows:

$$\begin{array}{rl} \text{Minimize} & \text{Path length or path vulnerability}, \\ \text{Subject to} & \text{Number of interfering cells} \leq 0. \end{array} \tag{2}$$

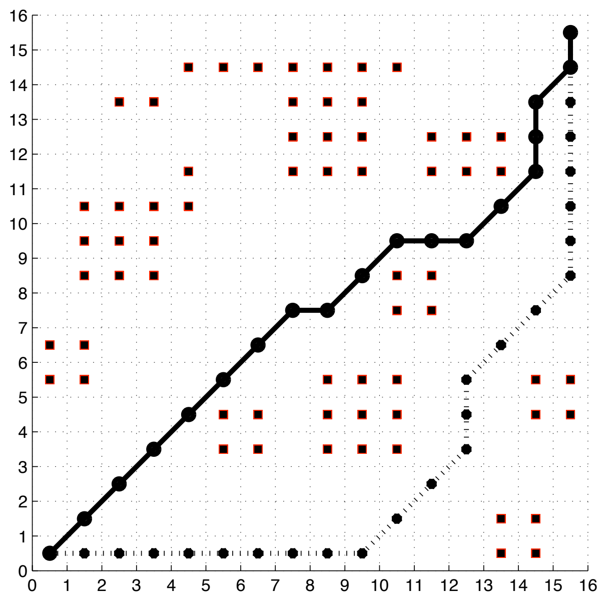
If, for any path, the number of interfering cells is greater than zero, we declare the path to be infeasible. Furthermore, in our proposed approach, we use the parameter-less single [8] or multi-objective [12] constraint handling technique, as the case may be, to handle infeasible and feasible solutions. In the tournament selection of comparing two solutions, if an infeasible solution is compared with a feasible solution, the latter always wins [8, 9]. On the other hand, if both solutions are feasible, the solution with a smaller objective value or having a better non-dominated front or crowding distance value wins. Finally, if both solutions are infeasible, the one with a smaller count on interfering cells wins. For brevity, we do not present any simulation results here, but state that this extended approach works well for cases with sparse obstacles, but fails to find a feasible path in cases with a highly dense set of obstacles in a single objective study. It has been observed that a better technique is to penalize the objective(s) with the count of interfering cells. Initial populations usually have infeasible solutions but the drive towards minimizing penalized functions eventually brings the number of interfering cells to zero. Based on these initial proof-of-principle studies, we use the penalty function approach for the rest of the paper here.

4.2 Single-objective Path Planning

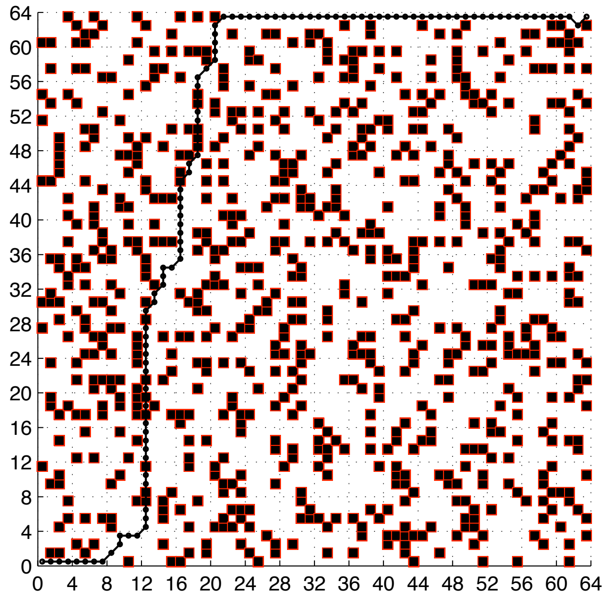
We have used integer-coded representation scheme described in Section 2.3 to code a path. In this study, only path length or the path vulnerability is minimized. The number of interfering cells is used as a constraint.

In Figure 3 we have shown two single-objective optimization runs – one with path length as an objective and the other with path vulnerability – for two obstacle scenarios. In the sparse obstacle scenario on a 16×16 grid (Figure 3(a)), both paths are shown. The GA is run with 400 population members and run for 600 generations. A crossover probability of 0.9 with a distribution index of 5 and a mutation probability of 1/16 and a distribution index of 20 are used. It is interesting to observe that the minimum path length solution (solid line) traverses almost in a straight line from start to finish. On the other hand, the minimum path vulnerability solution (dotted line) takes a somewhat longer path but traverses through a region with no obstacles. The trade-off between these two solutions is clear from this example. Interestingly, the same problem will be solved using a multi-objective optimization algorithm (described in the next subsection) involving both path length and path vulnerability as objective functions and two very similar extreme solutions are obtained by the proposed multi-objective optimization algorithm.

However, when the same algorithm is run for a larger grid size and more dense obstacle scenario having 64×64 grid and 19% cells covered with obstacles (shown in Figure 3(b)), the single-objective GA (with path length as an objective) fails to find a valid path and the best



(a) Sparse obstacle scenario in a 16 × 16 grid. Both solutions are feasible. The squares at some cells represent obstacles.



(b) Dense obstacle scenario in a 64 × 64 grid having 791 obstacles. The single-objective solution collides with 13 obstacles. The final infeasible path is shown.

Figure 3: Single-objective path planning results: (a) Single-objective algorithm is run twice, once with path length and next with path vulnerability as objectives, (b) Single-objective algorithm fails to find any obstacle-free path in the dense obstacle scenario.

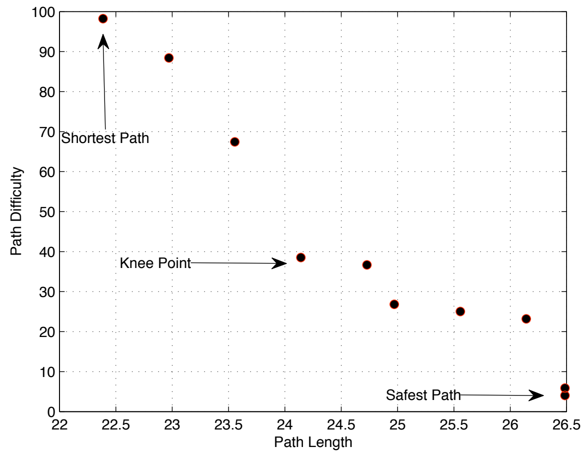
solution obtained by it collides with 13 obstacles. In this case, 500 population members are evolved for 500 generations and still no feasible solution is found. The same problem is solved later (in Section 7) with a multi-objective formulation and as shown in Figure 14(a), feasible paths are found that do not collide with any of the 791 obstacles.

Based on these two simulation studies, we conclude that the proposed single-objective GA is unable to perform well on large grid size and dense obstacle cases, whereas the multi-objective GA with both path length and path vulnerability is able to find a feasible path in different grid sizes and on both sparse and dense obstacle scenarios. Since multi-objective GAs maintain a diversity of solutions, the genetic operations are able to create new and better solutions, but since in a single-objective GA there is a single goal, the diversity is lost too quickly to find a feasible solution in the case of the dense obstacle problem. This emphasizes the usefulness of a multi-objective GA for handling complex path planning problems.

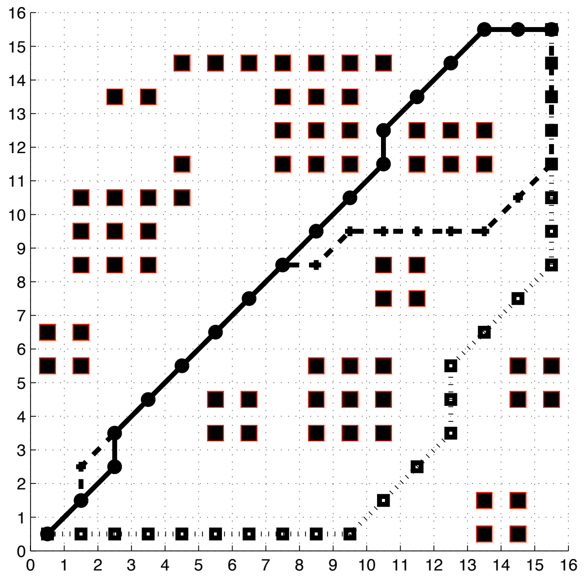
4.3 Multi-objective Path Planning

We modify the elitist non-dominated sorting genetic algorithm or NSGA-II algorithm [12] for this purpose. For details of NSGA-II, readers are encouraged to refer to the original NSGA-II study [12]. Here we describe the procedure briefly.

Initially, a population is initialized randomly. Thereafter, the population members are sorted based on their non-domination level in the population. In the so-called non-dominated sorting, the first front members are non-dominated members of the entire population. The second front members are those that are non-dominated population members for which first front members are excluded, and so on. Each individual in a front is assigned a crowding distance based on the distance of the neighboring solutions from it. A solution has a large crowding distance if its nearest neighbor lies far away from it in the objective space. In a tournament of comparing two



(a) Trade-off front obtained with the modified NSGA-II.



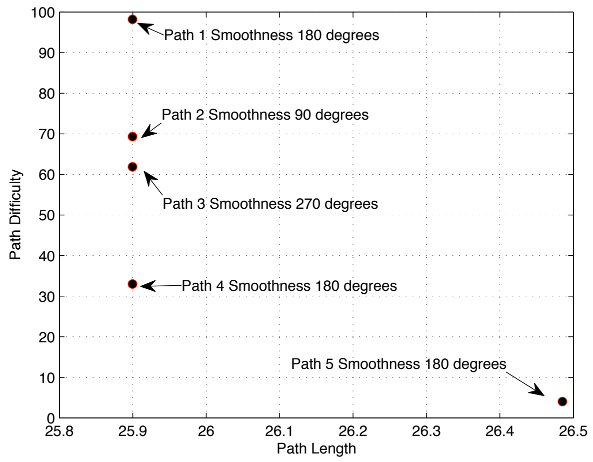
(b) Three paths – shortest, least vulnerable, and an intermediate path – are shown on a 16 × 16 grid.

Figure 4: Results from multi-objective optimization study are shown.

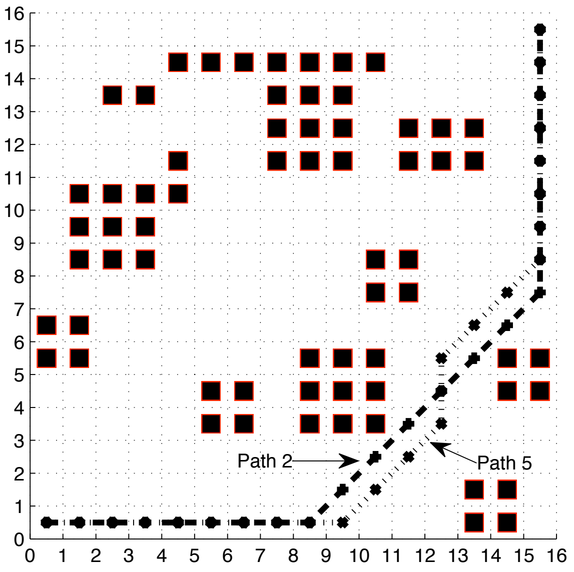
solutions, a solution is considered better if it lies on a better front or has a larger crowding distance value (thereby indicating an isolated solution). After the mating pool is created, they are used to create a new population using crossover and mutation operator exactly the same way as they are done in the case of single-objective GA. In the NSGA-II approach, both parent and newly created populations are merged and the combined population is sorted for their non-domination level again. Population members lying on the better fronts are chosen one at a time till the new population cannot accommodate any new front. To maintain the population size, all members of the last front, which could not be accommodated as a whole, are compared for their *crowding distance* values — a measure of emptiness in a solution's vicinity in the objective space [12] — and the ones having the largest crowding distance values are selected. This process is continued till the termination condition is met.

In the path planning problem, we have used both path length and path vulnerability as two objectives. The number of interfering cells is used as a constraint and is handled using the penalty function approach. To introduce the path smoothness as a criterion to prefer smooth paths, we modify the above NSGA-II's selection scheme somewhat. For two paths being compared in a tournament or in choosing the final front members, solutions of the same rank are checked for their smoothness values. The solution with a better smoothness wins. If two solutions being compared have an identical non-domination rank and identical smoothness value, then the crowding distance value is checked. We introduce the path smoothness criterion in another way to emphasize preferred paths. In the post-processing step, if two or more paths lie on the same point in the objective space (length-vulnerability trade-off front), path smoothness criterion is used to choose the single solution having the largest smoothness value.

We now discuss the results obtained with this multi-objective optimization algorithm on the sparse obstacle scenario presented in Figure 3(a). Here too, we use the integer-coding representation and a population of size 400 and run the modified multi-objective NSGA-II algorithm for 600 generations to obtain trade-off solutions. The trade-off solutions are shown in Figure 4(a). The range of path vulnerability and path length values obtained by this study shows that a diverse set of solutions are found by the proposed multi-objective optimization algorithm. Figure 4(b) shows



(a) Trade-off solutions obtained by three-objective study.



(b) Smoothest and the least difficult path are shown on a 16×16 grid.

Figure 5: Three-objective optimization results with smoothness as the third objective.

three paths picked from the obtained trade-off frontier. The solid line represents the shortest path, the dotted line shows the least vulnerable path and the dashed line shows an intermediate path. It seems that the chosen intermediate path makes a good compromise between the two extreme solutions (not too long and not too close to obstacles) and can be a preferred solution for an application. Interestingly, the shortest and least difficult paths are similar to those obtained by the single-objective optimization study in Figure 3(a).

4.3.1 Three-objective Optimization

In this subsection, we compare our above two-objective optimization study on path length and path vulnerability coupled with a third objective of smoothness as a decision-making aid with three objectives. We consider the same sparse obstacle scenario as considered above and use the integer-coding scheme to find optimal paths. Again, a population of size 400 is used for 600 generations. Figure 5(a) shows the obtained three-dimensional trade-off frontier projected on a two-dimensional plot. The path smoothness values are mentioned against each solution. Only five solutions are obtained, of which four solutions have an identical path length, but having differing path vulnerability and path smoothness values. The smoothest path and the safest path obtained are also shown in Figure 5(b).

With three objectives, more population members are non-dominated to each other. Thus, in a NSGA-II population, there is less room for storing new solutions. Moreover, the NSGA-II's crowding operator does not work well in three or more dimensions, as neighbors of a solution in one objective may not be neighbors in another objective in a higher dimensional objective space. These difficulties of NSGA-II has been reported elsewhere [16] and shows up here as well. Thus, we conclude from this comparative study that three-objective consideration of the path planning problem is not adequate. Henceforth, we use the two-objective modified NSGA-II procedure in which the path smoothness criterion is used as a decision making aid.

Having established the fact that (i) obstacle avoidance constraint handling is better through the penalty function approach, (ii) multi-objective optimization is better than single-objective path planning approach, and (iii) a two-objective approach coupled with path smoothness as a decision-making aid is a better approach than considering all three objectives in the optimization problem, we are now ready to perform a parametric study in arriving at reasonable values for the genetic parameters.

5 Parametric Study

The effect of various GA parameters on the path planning problem for the integer-coded representation scheme is studied here. The parameters considered in this study are as follows:

- Population size (N)
- Number of generations ($t_{\max}$)
- Crossover rate (p_c)
- Crossover distribution index (η_c)
- Mutation rate (p_m)
- Mutation distribution index (η_m)

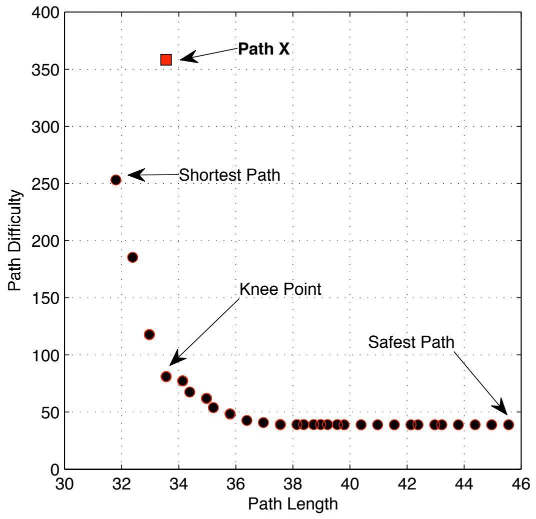
For the parametric study, we consider a particular obstacle placement shown in Figure 6(b). This scenario was the most complex scenario considered in an earlier path planing study [5]. A performance measure based on the hypervolume measure [37] is considered here to compare different trade-off fronts obtained by different parameter settings. To consider the effect of multiple runs, we define here a measure of likelihood for optimality, $Lopt_\zeta$, defined as follows. A similar measure was also defined in [32], but the metric did not account for trade-off fronts for a multi-objective scenario. To evaluate $Lopt_\zeta$, for a given grid size (n) and obstacle placement, first, we run and obtain H trade-off fronts by applying the algorithm H times from different initial populations. Thereafter, all H fronts are collected, dominated solutions are eliminated, and a reference hypervolume measure HV_{ref} is computed. Then, the number of individual runs (h) that have at least ζ% of the reference hypervolume is noted. The measure $Lopt_\zeta$ is defined as follows:

$$Lopt_\zeta = 100\frac{h}{H}. \tag{3}$$

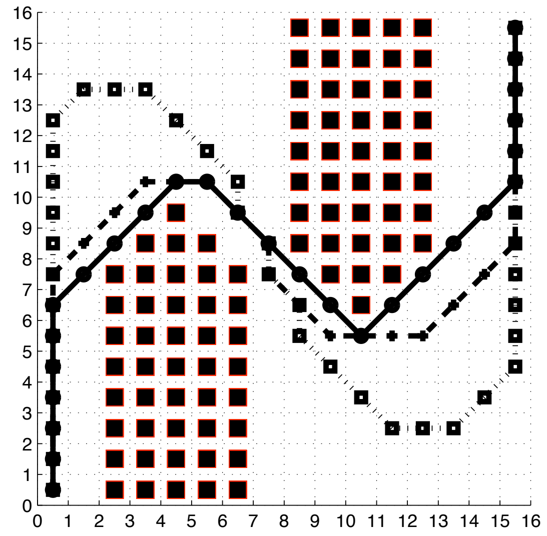
In all our simulations here, we have chosen $\zeta = 95$. Hence a value of $Lopt_{95} = 90$ means that in 90% of the runs have a hypervolume value that is more than 95% of the reference hypervolume.

The obstacle profile is as shown in Figure 6(b) along with three paths obtained by the integer-coding representation scheme. The two-objective NSGA-II is used with the path smoothness criterion used as a decision-making aid. The paths correspond to minimum path length, minimum path vulnerability, and an intermediate compromised solution. The trade-off frontier is shown in Figure 6(a). It is interesting to note that how the minimum path length solution nudges close to obstacles to reduce path length and how the minimum path vulnerability solution is keeping the path away from the obstacles to ensure safety. The intermediate solution is a compromise between these two extreme solutions. It is clear that the compromised solution lies on a 'knee' region of this trade-off front. When there exists a knee point on a trade-off frontier, it is usually a preferred solution [14], as there is not enough motivation to move away from the knee point due to unfavorable trade-offs.

The solution reported in [5] is marked on Figure 6(a). The study did not show multiple paths clearly with interconnecting grids, however, we extract the minimum path length configuration from the reported figure and compute path vulnerability using our computation method. Figure 6(a) shows that the earlier reported solution (indicated by 'Path X') is dominated by our



(a) Trade-off front obtained using two-objective NSGA-II. 'Path X' corresponds to solution found elsewhere [5].



(b) Obstacle arrangement and three obtained paths.

Figure 6: Obstacles on a 16 × 16 grid and obtained results for the parametric study.

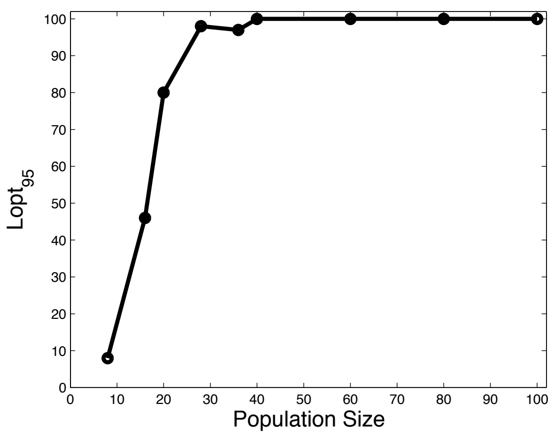
minimum path length solution and a couple of other solutions, thereby indicating superiority of our proposed procedure.

We now discuss the results of the parametric study. First, we present how the measure $Lopt_\zeta$ changes with the population size in Figure 7(a). While population size is varied in the range [8,100], the maximum number of generations is always kept fixed to 500. The other parameters are kept at their standard values mentioned before. Like most other successful GA studies [15], the figure concludes that below a critical population of size around 40, the proposed method does not have enough population members to its operators to make the method robust over 100 runs. However, when 40 or more population members are used, the algorithm works quite reliably.

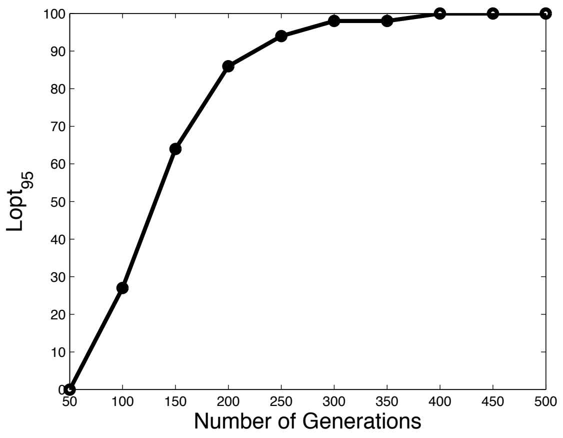
To be safe, we fix the population size to 60 and perform a parametric study on the maximum number of generations, by varying it in the range [50,500]. Figure 7(b) shows that the $Lopt_\zeta$ stabilizes at around 300. Based on this study, we fix the number of generations to 300 for the rest of the parametric study.

Next, we vary the crossover rate within [0,1] and observe from Figure 8(a) that a high crossover rate is more successful. This observation is in tune with existing GA studies [21, 31]. Based on this study, we set $p_c = 0.9$ for the rest of the parametric study. Figure 8(b) shows the effect of crossover distribution index η_c associated with the SBX operator on $Lopt_\zeta$. Over a range of [5,20], the effect of η_c is neutral and performance is almost equally good for any value. Thus, we choose $\eta_c = 10$ for the rest of our simulations.

Next, we perform a similar study with the mutation operator. Figure 9(a) indicates that too small or a too large mutation rate is not adequate for the proposed method. This agrees with another parametric study [11]. A value of p_m in the range [0.05,0.15] performs the best. Based on this study, we set $p_m = 0.0625$. For a real-parameter optimization with polynomial mutation operator, the upper bound on the recommended p_m is the reciprocal of the number of variables. With 16 grids along columns, there are 15 variables and the upper bound on p_m is chosen to be 1/15 or 0.067. Thus, our experimental finding of $p_m = 0.0625$ is not far from this recommended value. Also Figure 9(b) shows that a higher η_m performs better. We set $\eta_m = 20$ for the rest of our study.

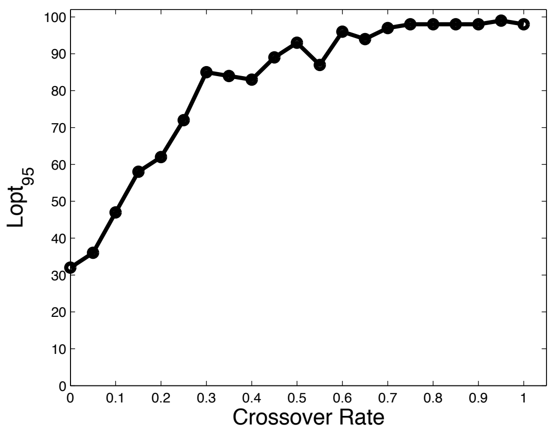


(a) Effect of population size.

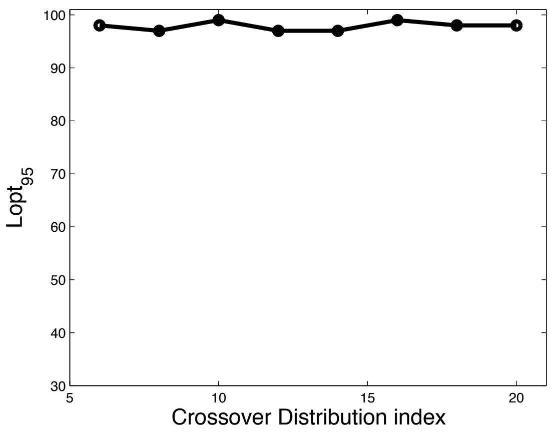


(b) Effect of number of generations.

Figure 7: Parametric study on population size and number of generations.

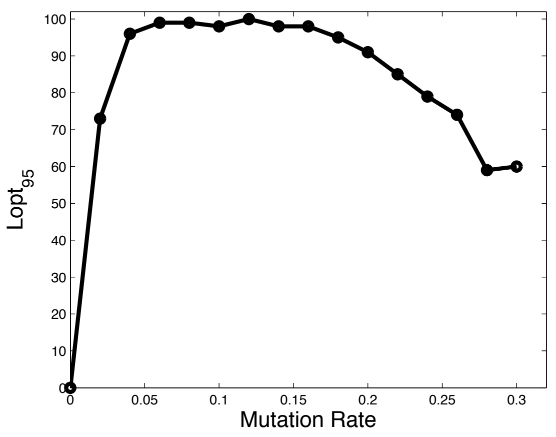


(a) Effect of crossover rate.

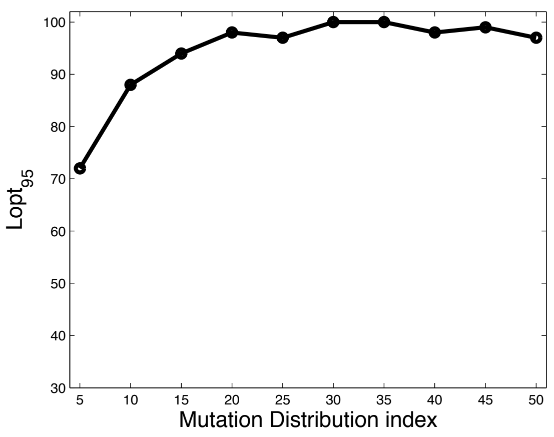


(b) Effect of crossover distribution index.

Figure 8: Parametric study on crossover rate and SBX distribution index.

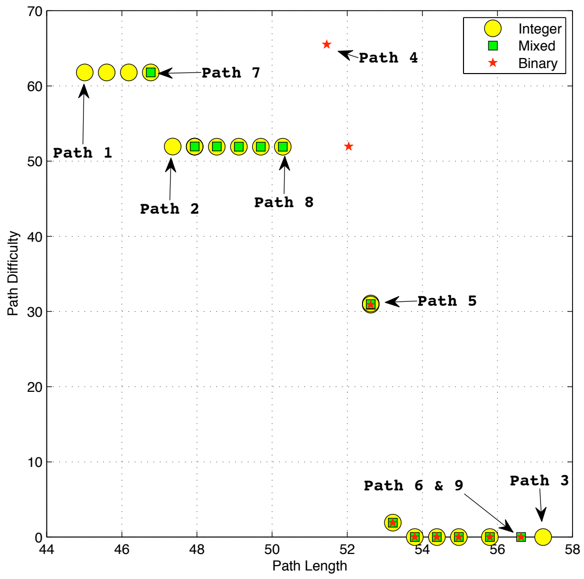


(a) Effect of mutation rate.

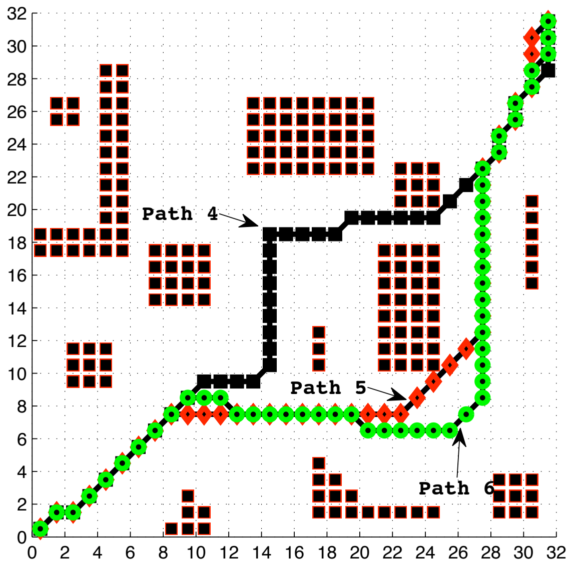


(b) Effect of mutation distribution index.

Figure 9: Parametric study on mutation rate and polynomial distribution index.



(a) Trade-off points obtained using all three representation schemes. Integer-coded representation scheme perfomrms the best.



(b) Obstacles on a 32 × 32 grid and few solution paths obtained using binary-coded representation scheme.

Figure 10: Results with binary-coded representation scheme.

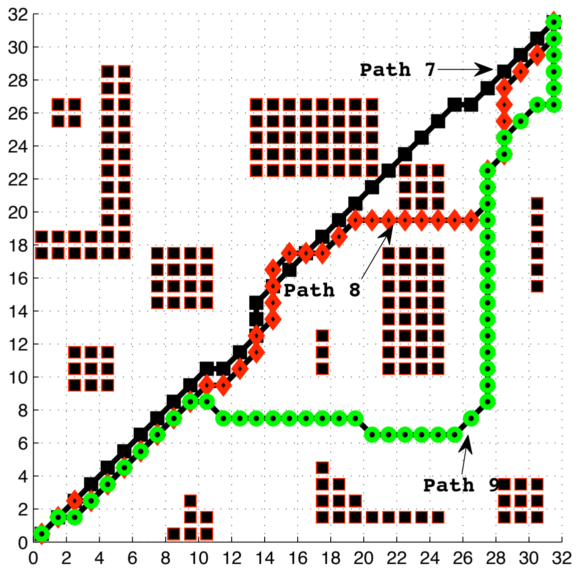
The above parametric study on a single scenario supports our earlier understanding of a good setting of these parameters. Based on our experience on other problems and supported by the above parametric study, we recommend the following values for the NSGA-II parameters (n is the size of the grid):

- Minimum population size (N) $= 10n$
- Minimum number of generations ($t_{\max}$) $= 300$
- Crossover rate (p_c) $= 0.9$
- Crossover distribution index (η_c) $= 10$
- Mutation rate (p_m) $= 1/(n-1)$
- Mutation distribution index (η_m) $= 20$

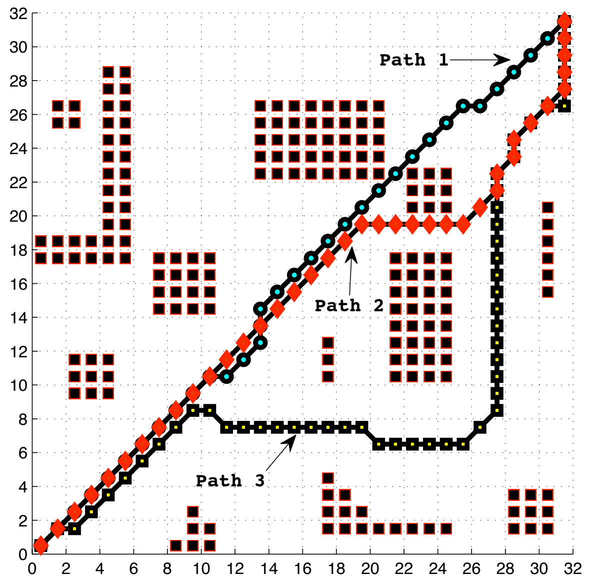
Higher population sizes and the number of generations may be essential to handle a higher grid size problem and a more dense obstacle scenario than that used in the above parametric study.

6 Comparison of Three Representation Schemes

In all the above simulations, we have used the integer-coded representation scheme. In this section, we use the other representation schemes, namely, the binary-coded and mixed-coded representation schemes and compare all three schemes. To compare, here we choose a 32×32 obstacle arrangement shown in Figure 10(b). We choose a population of size 500 and run the bi-objective NSGA-II for 800 generations. Other parameters are set according to the above parametric study. Figure 10 is obtained with a binary-coded representation scheme. The bit-wise mutation probability for the binary string of $p_m = 0.00325$ is used here. The best trade-off front obtained in 10 different runs is shown in Figure 10(a) with a small star. The minimum



(a) Obstacles on a 32 × 32 grid and few solution paths obtained using mixed-coded representation scheme.



(b) Obstacles on a 32 × 32 grid and few solution paths obtained using integer-coded representation scheme.

Figure 11: Results with mixed and integer-coded representation schemes.

path distance solution, minimum path vulnerability solution and an intermediate compromised solution are shown in Figure 10(b). Corresponding objective vectors are also marked in the adjoining figure. Figure 10(b) shows how the intermediate solution negotiates the distance and safety.

Figure 11(a) also shows the trade-off objective vectors using the mixed-coded representation scheme. Identical parameter values, as they were used with the binary-coded representation scheme, are chosen here. For the mixed-coding scheme, $p_m = 1/16$ is used for integer variables and $p_m = 0.052$ is used for the binary substring. The obtained trade-off frontier and the three paths shown in the figure are better than that obtained with the binary-coded representation scheme. Similarly Figure 11(b) shows the results obtained with the integer-coded representation scheme. Parameters identical to those found in the parametric study are used here. The paths are better than the mixed-coded representation scheme.

Each of the three representation schemes is run for 10 times from different initial populations. All trade-off frontiers are collected and the nadir point of the combined front is recorded. Thereafter, the nadir point is used as the reference point and the hypervolume measure is computed for each frontier. Figure 12 shows the hypervolume value of each of the 10 runs used with each of the three representation schemes. It is clear that largest hypervolume value occurs for the integer-coded representation scheme. The binary-coded scheme performs the worst. Table 1 tabulates the $Lopt_\zeta$ values for three methods with different ζ values. The table agrees with our observation that the integer-coded representation is better than the mixed-coded representation, which is in turn better than the binary-coded representation scheme on the chosen problem.

7 Path Planning Under Extreme Conditions

An extreme test of a path planning algorithm is to try it to solve scenarios having a large grid size and a dense set of obstacles. When adequate spaces are available, multi-objective optimization algorithms can be used to find a set of trade-off solutions. However, when there more obstacles

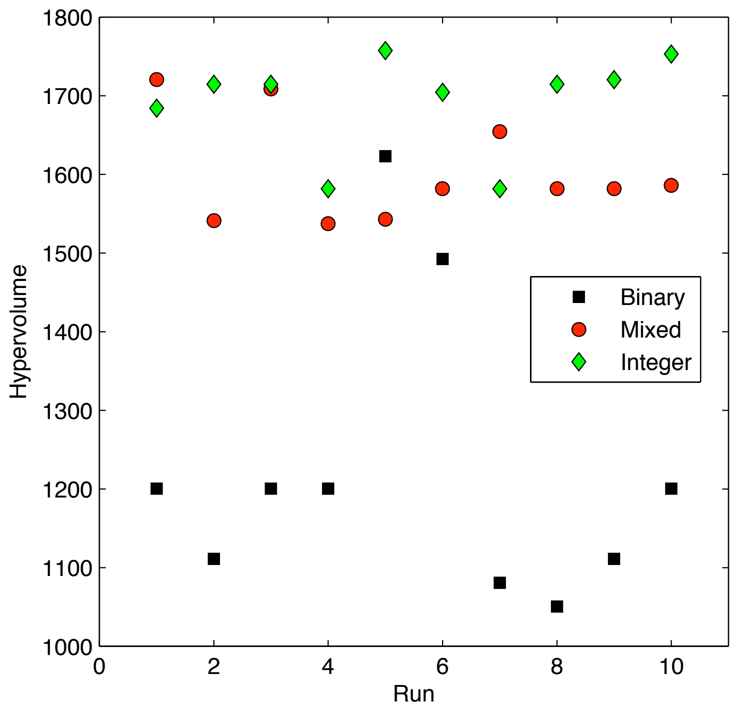


Figure 12: Hypervolume values for all 10 runs for each of the three representation scheme are shown. Higher the volume, the better is the performance.

Table 1: $Lopt_\zeta$ values are shown for each of the three representation scheme on problem shown in Figure 10(b).

Coding	$Lopt_\zeta$ in %							
scheme	ζ =95	90	85	80	75	70	65	60
Binary	0	10	10	20	20	20	60	90
Mixed	20	40	100	100	100	100	100	100
Integer	80	80	100	100	100	100	100	100

than the available space in a grid, finding a single feasible path is often a challenging task. In this section, we create increasingly higher grid size and increasingly dense obstacle scenarios in which obstacles are placed randomly on a two-dimensional grid. However, we make sure that there exists at least one feasible path from start to the destination point. To achieve this, we first generate a random valid path on the entire grid. Then, we randomly create obstacles with p_0 probability of occurrence of obstacle in every cell except those lying on above generated path. Hence, a probability of $p_0 = 1$ would mean that all grid cells have obstacles on them except the cells lying on the random path initially created. Such a problem would be a very difficult path planning problem to solve. To make the problems of different size, we use grid size that is varied to 8, 16, and 32. The probability p_0 is varied from 0.1 to 1, as tabulated in Table 7. For all problems, the integer-coded representation scheme is used. For each problem, a population of size 200 is used and the NSGA-II is run for 500 generations. Other parameters are set according to the outcome of the parametric study presented in the previous section. For each problem, 100 runs are taken and the percentage of successful runs – finding a feasible path from start to destination point, mean minimum path length obtained, and the median of generation numbers in which the first feasible path is obtained are noted. The first entry indicates the likelihood of finding a feasible path, the second entry denotes the ability to further minimize path length for feasible path while the third entry denotes how quickly the first feasible path is found on an average. There are several observations to be made from the table. First, as the grid size increases, the success percentage reduces, however the proposed NSGA-II is able to still find the feasible path in all problems with 8×8 and 16×16 grids. The number of generations needed to find a feasible path is also small. Since a population of size 200 is used here, the 32×32 grid problems are too large for the chosen population size. A larger population may able to solve scenarios with a higher grid size and having a higher density of obstacles. The second matter to note is that with 16×16 grid problems, the proposed methodology is able to find a feasible path in a problem having a density of obstacles as large as $233/(16 \times 16)$ or 91%. This is a remarkable result. The third aspect to note is that as the density of the obstacles increase, the problem gets harder to solve and more generations are needed to locate a feasible path.

Figure 13 shows a few of the obtained solutions. Figure 13(d), for example, shows how in a large grid size and among a large density of obstacles, the proposed methodology is able to locate a feasible path, a task which may be difficult, even for a human expert.

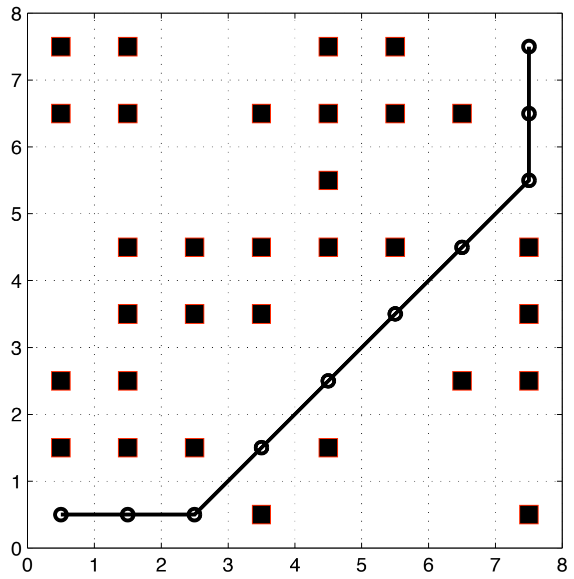
To investigate how the proposed bi-objective procedure will fair on larger grid size problems, finally we apply our methodology on a few 64×64 and 128×128 grid problems. Figure 14(a) shows a couple of runs and the obtained feasible paths in both cases. Clearly, when 4,979 obstacles are scattered randomly on a two-dimensional grid (as shown in Figure 14(b), it is impossible to obtain a feasible path manually. Although many other solutions may be possible in this case, our proposed algorithm is able to find one feasible path.

8 Conclusions

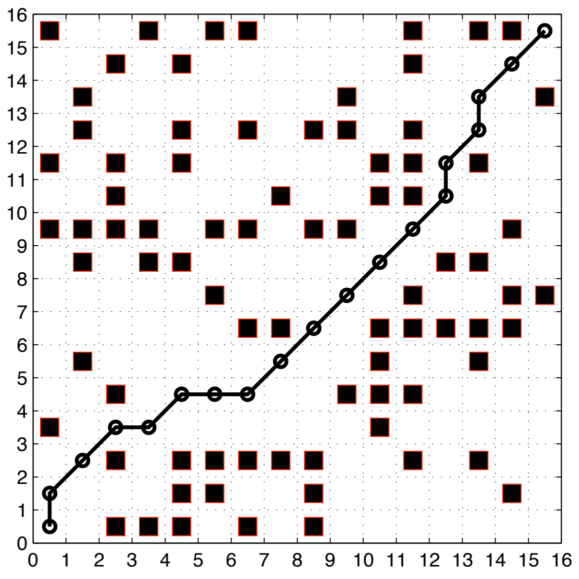
In this paper, we have dealt with several issues of a two-dimensional path planning problem having various complexities. First, we have suggested three path representation schemes for optimal path planning tasks to be solved by using a multi-objective genetic algorithm. Of the three schemes, the integer-coded gene representation scheme that directly codes the relative movements of a vehicle from one column to the next has been found to produce best results. Second, it has been observed that instead of considering three different criteria – path length, path vulnerability, and path smoothness – as objectives, the use of the first two as objectives and the third as a decision-making aid is a better option. Third, the above multi-objective consideration has been found to be better than a single-objective treatment of say path length or path vulnerability. Fourth, compared to existing studies, the proposed approach has been shown to be more accurate in

Table 2: Path planning under different volumes of dense obstacles.

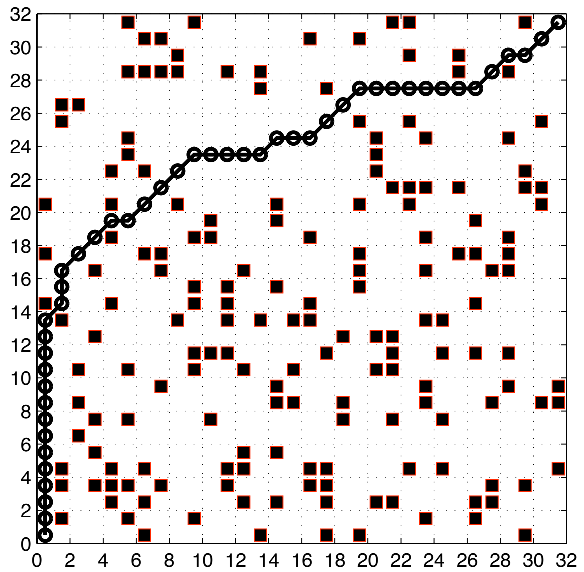
Grid size	Obstacle density, p_0	# of obstacles	Success percentage	Median generations	Mean min. path length
8×8	0.1	4	100	2	9.90
	0.2	10	100	2	10.49
	0.3	15	100	2	10.49
	0.4	20	100	2	10.49
	0.5	26	100	5	11.07
	0.6	31	100	4	11.07
	0.7	40	100	5	11.07
	0.8	43	100	6	11.07
	0.9	50	100	7	11.07
	1.0	54	100	7	11.07
16×16	0.1	24	100	2	21.21
	0.2	47	100	5	22.14
	0.3	65	100	9	23.3
	0.4	80	100	18	22.34
	0.5	115	100	28	22.37
	0.6	150	100	22	22.57
	0.7	164	100	25	22.3
	0.8	286	100	40	22.13
	0.9	214	100	41	22.48
	1.0	233	100	46	22.4
32×32	0.1	81	100	6	47.08
	0.2	173	100	16	53.4
	0.3	253	97	38	53.72
	0.4	386	85	73	53.61
	0.5	470	36	230	54.38
	0.6	556	74	144	53.64
	0.7	666	41	295	53.9
	0.8	767	52	298	54.31
	0.9	870	39	378	54.13
	1.0	967	30	339	54.19



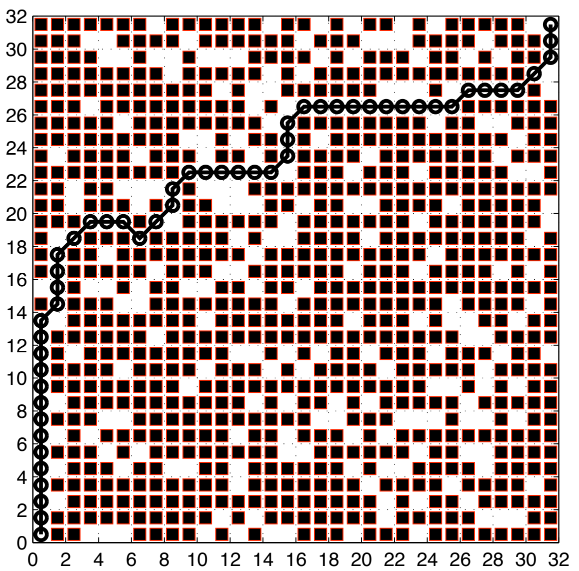
(a) Grid 8×8, $p_0 = 0.6$, 31 obstacles.



(b) Grid 16×16, $p_0 = 0.4$, 80 obstacles.

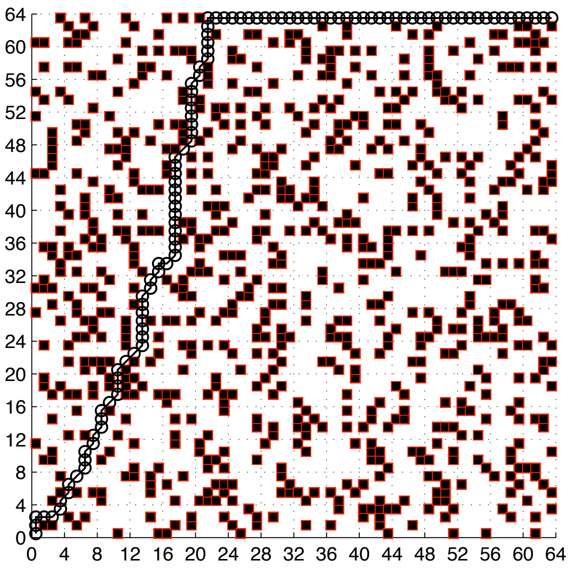


(c) Grid 32×32, $p_0 = 0.2$, 173 obstacles.



(d) Grid 32×32, $p_0 = 0.8$, 767 obstacles.

Figure 13: Feasible paths for cases shown in Table 7.


(a) Problem in a 64 × 64 grid having 791 obstacles.

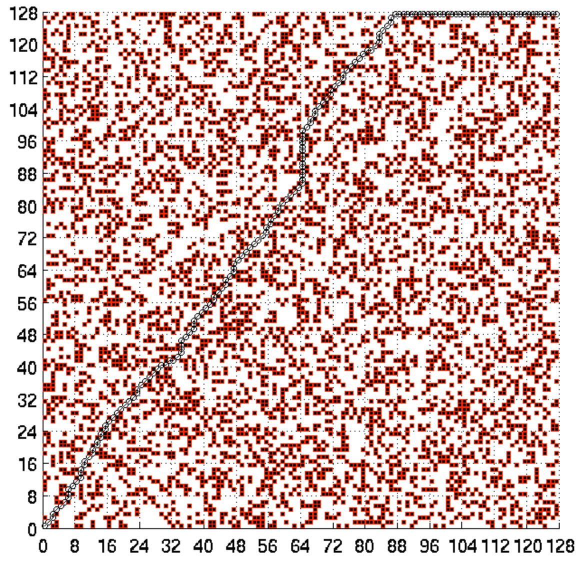

(b) Problem in a 128 × 128 grid having 4,979 obstacles.

Figure 14: Feasible paths for large grid size problems obtained using the proposed bi-objective procedure.

finding feasible and trade-off paths. Additionally, the proposed method has been demonstrated to work on large-sized grids and having a dense set of obstacles (as high as 91% of the space). While the existing past studies limited their studies to small-size grids, here we have found feasible paths in grids as large as 128 × 128 and having as large as 4,979 obstacles. Some of the scenarios are intimidating to look for a suitable path visually and have also been found to be difficult to solve using conventional techniques or using a single-objective genetic algorithm. The inherent diversity preserving ability of a multi-objective GA enables finding feasible solutions in such difficult scenarios.

Here, we have also demonstrated how through a systematic parametric study and a careful consideration of deferent aspects of the problem, an efficient algorithm can be evolved for solving a complex path planning task. This work can be extended to include non-monotonic paths along both axes which will make it a more practical scheme for real-world applications. Three-dimensional path planning can also be tried with minor modifications in the proposed gene structures. Nevertheless, extensive simulation results of this study indicates that the current multi-objective technique is a reliable tool for a point to point, off-line path planning task and must be pursued further.

Acknowledgments

The study is supported by Department of Science and Technology, Government of India under SERC-Engineering Sciences scheme (No. SR/S3/MERC/091/2009).

References

[1] Ahuja, N., Chuang, J.: Shape representation using a generalized potential field model. Pattern Analysis and Machine Intelligence, IEEE Transactions on **19**(2), 169–176 (1997)

[2] Al-Sultan, K.S., Aliyu, M.D.S.: A new potential field-based algorithm for path planning. Journal of Intelligent & Robotic Systems **17**(3), 265–282 (2010)

[3] Bisse, E., Bentounes, M., Boukas, E.K.: Optimal path generation for a simulated autonomous mobile robot. Autonomous Robots **2**(1), 11–27 (1995)

[4] Burchardt, H., Saloman, R.: Implementation of path planning using genetic algorithms on mobile robots. In: Proceedings of the World Congress on Evolutionary Computation (WCCI-2006), pp. 1831–1836 (2006)

[5] Castillo, O., Trujillo, L.: Multiple objective optimization genetic algorithms for path planning in autonomous mobile robots. International Journal of Computers, Systems and Signals **6**(1), 48–63 (2005)

[6] Choset, H.: Sensor based motion planning: The hierarchical generalized Voronoi graph. Ph.D. thesis, Citeseer (1996)

[7] Connolly, C., Burns, J., Weiss, R.: Path planning using laplace's equation. In: Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on, pp. 2102–2106. IEEE (1990)

[8] Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering **186**(2–4), 311–338 (2000)

[9] Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester, UK (2001)

[10] Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems **9**(2), 115–148 (1995)

[11] Deb, K., Agrawal, S.: Understanding interactions among genetic algorithm parameters. In: Foundations of Genetic Algorithms 5 (FOGA-5), pp. 265–286 (1999)

[12] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)

[13] Deb, K., Goyal, M.: A robust optimization procedure for mechanical component design based on genetic adaptive search. Transactions of the ASME: Journal of Mechanical Design **120**(2), 162–164 (1998)

[14] Deb, K., Gupta, S.: Understanding knee points in bicriteria problems and their implications as preferred solution principles. Engineering Optimization (in press)

[15] Deb, K., Reddy, A.R., Singh, G.: Optimal scheduling of casting sequence using genetic algorithms. Journal of Materials and Manufacturing Processes **18**(3), 409–432 (2003)

[16] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. In: A. Abraham, L. Jain, R. Goldberg (eds.) Evolutionary Multiobjective Optimization, pp. 105–145. London: Springer-Verlag (2005)

[17] Elshamli, A., Abdullah, H.A., Areibi, S.: Genetic algorithms for dynamic path planning. In: Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE-04), pp. 677–680 (2004)

[18] Ge, S.S., Cui, Y.J.: Dynamic motion planning for mobile robots using potential field method. Autonomous Robots **13**(3), 207–222 (2002)

[19] Gerke, M.: Genetic path planning for mobile robots. In: American Control Conference, 1999. Proceedings of the 1999, vol. 4, pp. 2424–2429. IEEE (1999)

[20] Glasius, R., Komoda, A., Gielen, S.: Neural network dynamics for path planning and obstacle avoidance. Neural Networks **8**(1), 125–133 (1995)

[21] Goldberg, D.E.: Genetic Algorithms for Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley (1989)

[22] Hwang, Y., Ahuja, N.: Gross motion planninga survey. ACM Computing Surveys (CSUR) **24**(3), 219–291 (1992)

[23] Kanayama, Y., Hartman, B.: Smooth local-path planning for autonomous vehicles. International Journal of Robotics Research **16**(3), 263 (1997)

[24] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. International Journal of Robotics Research **5**(1), 90 (1986)

[25] LaValle, S.: Planning algorithms. Cambridge Univ Pr (2006)

[26] Lozano-Pérez, T., Wesley, M.: An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM **22**(10), 560–570 (1979)

[27] Murrieta-Cid, R., Tovar, B., Hutchinson, S.: A sampling-based motion planning approach to maintain visibility of unpredictable targets. Autonomus Robots **19**(3), 285–300 (2005)

[28] Oriolo, G., Ulivi, G., Vendittelli, M.: Fuzzy maps: a new tool for mobile robot perception and planning. Journal of Robotic Systems **14**(3), 179–197 (1997)

[29] Pratihar, D., Deb, K., Ghosh, A.: Fuzzy-genetic algorithms and time-optimal obstacle-free path generation for mobile robots. Engineering Optimization **32**, 117–142 (1999)

[30] Sauter, J., Matthews, R., van Dyke Parunak, H., Brueckner, S.: Evolving adaptive pheromone path planning mechanisms. In: Proceedings of the First International Joint Conference on Autonomous agents and Multiagent Systems: part 1, pp. 434–440. ACM (2002)

[31] Schaffer, J.D., Caruana, R.A., Eshelman, L.J., Das, R.: A study of control parameters affecting online performance of genetic algorithms. In: Proceedings of the International Conference on Genetic Algorithms, pp. 51–60 (1989)

[32] Sugihara, K.: Measures for performance evaluation of genetic algorithms. In: Proc. 3rd. joint Conference on Information Sciences, pp. 172–175. Citeseer (1997)

[33] Sugihara, K., Smith, J.: Genetic algorithms for adaptive planning of path and trajectory of a mobile robot in 2D terrains. IEEE Transactions on Information and Systems **82**(1), 309–317 (1999)

[34] Xiao, M., Michalewicz, Z.: An evolutionary computation approach to robot planning and navigation. Soft Computing in Mechatronics pp. 117–128 (2000)

[35] Xue, Q., Sheu, P.C.Y., Maciejewski, A.A., Chien, S.Y.P.: Planning of collision-free paths for a reconfigurable dual manipulator equipped mobile robot. Journal of Intelligent & Robotic Systems **17**(3), 223–242 (2010)

[36] Yang, S.X., Meng, M.: Real-time collision-free path planning of robot manipulators using neural network approaches. Autonomous Robots **9**(1), 27–39 (2000)

[37] Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation **3**(4), 257–271 (1999)