An Evolutionary Algorithm for Multi-Objective Path Planning Problem in Virtual Environments

Tuğcem Oral
Department of Computer Engineering
Middle East Technical University, 06800 Ankara, Turkey
tugcem.oral@gmail.com

***Abstract*—Path planning in virtual environments still remains as an issue while reflecting it to real world applications. In this paper, we present an evolutionary solution to multi-objective path planning (MOPP) problems and describe its behaviors in terms of diversity and optimality. These problems consist of generating valid paths for an agent from an initial position to a goal across a flat map of a terrain, represented by a two dimensional grid, with obstacles and different weighted cells. The algorithm execution results show that several optimal solutions can be found in polynomial time. This situation creates an alternative way for other problems which are intractable.**

***Index Terms*—Multi-objective, path planning, evolutionary algorithms.**

I. Introduction

MULTI-objective problems differ itself from classical (single-objective) problems via considering more than one criteria (objectives) at the same time while finding a solution. The task in a multi-objective problem is not to find an optimal solution for each objective but to find an optimal solution that simultaneously optimizes *all* objectives. And in most cases, there exist a set of optimal or non-dominated solutions rather than a single solution.

As looking from path planning perspective, single-objective path planning considers only one constraint (i.e. the shortest or the longest path) while finding a valid path from initial to goal position. However, multi-objective path planning (MOPP) comes into play with more than one constraints while deciding a path. The MOPP problem has been studied extensively by various researchers in the fields of optimization, information & communications in a network and route planning for traffic. For instance, the problem of finding a valid path to a target of an Unmanned Aerial Vehicle (UAV) which considers fuel consumption and getting the optimal path which might includes anti-aircraft risky radar regions could be given as an issue for MOPP.

Multi-objective path planners must be able to generate valid paths from initial to goal position, and in order to truly show a level of intelligence these paths must be *optimized* under the criteria as given with problem. A perfect and precise path planner, which finds the path if one exists, otherwise reports no path existence, is NP-hard [2]. It has been shown that a set of problems exist wherein the number of Pareto-optimal solutions is *exponential* which implies that any deterministic algorithm that attempts to solve it is also exponential in terms of runtime complexity at least in the worst case.

Even there exists many old research about MOPP algorithms, there still seems to be a lack of reported literature in evolutionary approaches in relation to the MOPP. In order to demonstrate a clearer picture of the advantages and disadvantages of evolutionary algorithms in optimization, this paper attempts to investigate a multi-objective evolutionary algorithm applied to MOPP.

The paper is organized as follows. Section II gives related literature. Section III presents background concepts of proposed algorithm. Section IV proposes and elaborates the multi-objective evolutionary algorithm for path planning. Experimental results are discussed in Section V. Finally, Section VI gives the conclusion and details future work.

II. Related Work

For virtual environments, there exist many proposed heuristic solutions to MOPP. Koneig et al. introduce a new incremental heuristic search algorithm, D* lite in [7]. This algorithm reuses search trees from previous searches to speed up the current search and finally finds cost-minimal paths for series of similar search problems. They mention that this method is faster than by solving each search problem from scratch.

Bukhari et al. comes up with an optimization technique for dynamic online path planning and optimization of the path [1]. It addresses the issues involved during path planning in dynamic and unknown environments cluttered with obstacles and objects. A simulated ant agent system is proposed using modified ant colony optimization algorithm for dealing with online path planning. It is compared with evolutionary techniques on randomly generated environments with different obstacle ratios and grid sizes. The proposed algorithm generates and optimizes paths in complex and large environments with several constraints.

Nasrollahy et al. proposes a particle swarm optimization algorithm as a multi-agent search technique, for path planning in dynamic and known environments in order to minimize total path planning time while avoiding the local optimums [8]. They create a small-scale model of search system moving goal position and obstacles. These obstacles are defined as circular shapes and agents get around of these obstacles. They try to optimize global best path through to the goal position. Although they mention about effectivity of proposed algorithm, they don't give concrete results and comparisons with other methods.

As well as heuristic approaches, there are several proposed evolutionary solutions for MOPP. A recent study by Pangilinan et al. [9] introduce an evolutionary algorithm for multi-objective shortest path problem. They draw the picture of their 2-D static (stable obstacles and target) environment as a graph. Initial population is created by randomly generated individuals where each has a random ordered path from initial position to goal position. They use binary tournament selection for mating. Strength Pareto Evolutionary Algorithm (SPEA2) evaluates fitness values of individuals and selects them for survival. They define density function of fitness evaluation to avoid from genetic drift. For genetic operators, they use one-point crossover and mutation. Their results show that their algorithm is a good alternative in finding a subset of efficient solutions for multi-objective shortest path problems when performance issues like complexity, diversity and nondominal optimal solutions become obstructions.

Dozier et al. gives a new selection methodology for MOPP in [6]. They introduce fuzzy tournament selection algorithm which combines fuzzy inference with tournament selection to select candidate solution paths. This selection is based on the euclidean distance from initial to goal position, the sum of the changes and the average change in the slope of a path.

Castillo et al. also worked on evolutionary algorithms for MOPP in [3]. They define a genetic offline point-to-point agent path planner which tries to find valid paths. They concentrate on two constraints which are path length and difficulty (each path has a difficulty which calculated from predefined weights) in their 2-D static grid environment. They compare their results with 90's papers and get better results.

A complete discussion of multi-objective evolutionary algorithms (MOEA) can be found in [5]. Also [4], gives a summary of current approaches in MOEA and emphasizes the importance of new approaches in exploiting the capabilities of evolutionary algorithms in multi-objective optimization.

III. Background

In our proposed solution to MOPP, we decide to use 2-D grid environment for working around. Given a grid $G = \{C, n\}$ where C is the set of cells and $n \in \mathbb{Z}^+$ is the size of this grid. Note that

$$C = \{c_{1,1}, c_{1,2}, ..., c_{n,1}, c_{n,2}, ..., c_{n,n}\}$$

has $n \times n$ cells. If initial cell of the path finder agent is $\varsigma = \sigma_0 = c_{1,1}$ and the goal cell is $\tau = \sigma_{k+1} = c_{n,n}$, a path p can be defined as a set of specified cells from initial to goal cell as follows:

$$p = \{\varsigma, \sigma_1, \sigma_2 = adj(\sigma_1), ..., \sigma_k = adj(\sigma_{k-1}), \tau = adj(\sigma_k)\}$$

where $i, j = \{1, 2, 3, ..., n\}$, $k = \{0, 1, 2, 3, ..., n \times n - 1\}$ and $\sigma_1 = adj(\varsigma) = adj(c_{1,1})$. k should get $n \times n - 1$ as maximum value that a valid path p should not pass through same cell twice. The $adj(\sigma_k)$ function finds the *available adjacent* of given cell σ_k and returns one of them randomly if finds more than one. Notice that a cell can has maximum eight adjacent, cardinal and intercardinal points. A cell $c_{i,j}$ is available for an agent if this cell is inside the grid ($1 \leq i, j \leq n$) and does not have any obstacle on it.

Each cell on the grid has the following properties:

$$c = \{x, y, w\}$$

x and y are the coordinates of the cell, respectively. w is the weight of the cell which $0 \leq w \leq 1$. The more weight a cell has, the more difficult to pass through for agent. If $w = 1$, that means this cell has an obstacle.

The obstacles and goal cell are preferred to be static and predefined before the execution of the algorithm. However, agent doesn't know the environment and it tries to find a path to goal cell. Weights of cells are randomly generated and does not change during the execution.

As a result of multi-optimality, we mentioned that there could be more than one optimal solution as a result. These results are paths, indeed. Set of results, or paths are represented like $P(\varsigma, \tau) = \{p_1, p_2, p_3, ..., p_m\}$. m is the number of the found solutions.

Optimality is a crucial concept for MOPP. We define the optimality vector $o(p)$ of a path p as the sum of optimalities of each cell on the path ($o(\sigma_i)$), that is

$$o(p) = \sum_{i=0}^{k} o(\sigma_i)$$

If all objectives are to be maximized, a path $p \in P(\varsigma, \tau)$ dominates a path $q \in P(\varsigma, \tau)$ iff $o(p) \geq o(q)$ and we could write $p \preceq q$. A path p is Pareto-optimal if it is not dominated by any other path and the set of non dominated solutions (paths) is called the Pareto-optimal set. The objective of the multi-objective path planning algorithm is to compute the set of non dominated solutions that is the Pareto-optimal set P of $P(\varsigma, \tau)$ with respect to $o(\sigma_i)$.

Like every evolutionary algorithm, we have a population which is a set of individuals in our solution. These individuals are represented by possible valid paths. Each individual has a path from initial to goal cell.

IV. Multi-Objective Evolutionary Algorithm

We can now get into the architecture of proposed multi-objective evolutionary algorithm (MOEA) to find paths between initial and goal cells. Simply the algorithm schema is given as follows:

As it can be seen, it is quite similar with a classic evolutionary algorithms. We first create the initial population, than evolve this population until a predefined iteration number. While evolving; at each generation, a new set of individuals are created. We use elitist replacement for survival selection, roulette-wheel selection for mating selection, one-point crossover and mutation as genetic operators. We evaluate the newish population with a fitness function and replace with old population. Finally, we could gather our optimal solutions, or paths. This process leads to the evolution of populations of individuals that are better suited to their environment than their ancestors, just as in natural adaptation. A complete discussion of multi-objective evolutionary algorithms can be found in [5]. We can now elaborate our each step deeply:

Algorithm 1 The MOEA

```
function findMultiObjectivePaths()
input grid, initialCell, finalCell
output listOfOptimalPaths
    population = initializePopulation();
    repeat
        newPopulation = defineNewPopulation();
        newPopulation.add(population.getElites());
        repeat
                parents = population.rouletteWheelSelection();
                crossover(parents);
                mutation(parents);
                newPopulation.add(parents);
        until newPopulationIsConstructed();
        evaluatePopulation(newPopulation);
        population = newPopulation;
    until MAX_ITERATION
end
```

A. Initial Population

The initial population, set of individuals S is initialized with m total individuals. Each individual's path is generated *randomly*. For each individual; the algorithm starts from initial cell $c_{1,1}$, gets available adjacent of corresponding cell with $adj(\sigma_{actual})$ and select one of the adjacent randomly. This process is iterated until goal cell is reached.

As given the method to create valid feasible paths, the problem of population size m remains as an issue. It is known that the population size increases exponentially with the number of objectives in multi-objective domain. Thus, there are two common options to overcome this problem. First one is using a large population, second one is making the population sizing dynamic. However, dynamic resizing remains as a challenge to MOEAs and does not yield precise results. So, Deb's [5] approximation chart for finding the minimum population size according to the number of objectives is used as a guide.

B. Fitness Evaluation

After initialization of first population and at the end of each evolution step, the individuals in actual population must be evaluated with fitness function. This evaluation is crucial to provide the evolution of the population. The evaluation of a population means evaluating every single individual and putting them into an order. We expect that the population evaluation value should increase in each step. As we consider *weights* and *number of cells on a path* as constraints, we use the following fitness function:

$$f(p) = \frac{1}{\left(\frac{\sum_{i=0}^{k} w_i}{k}\right)^2 + (k)^2}$$

Where k is the number of the cells in the path, $w_i$is the weight of a cell. We simply get the average of the weights of cells on the path and add its square with the square of number of the cells on the path. This result is divided with 1 to keep the fitness function between 0 and 1 to increment sensitivity of fitness value.

C. Selection for Mating

To breed new individuals and to use the operators borrowed from natural genetics, we must select parents first. We use *roulette-wheel selection* while selecting parents. After evaluating each individual as indicated above, slots are generated and individuals' fitness values are added cumulatively to these slots. Thus, bigger fitness evaluation owners have greater opportunity to be selected. This mechanism facilitates the population to evolve.

While selecting two individuals, it is constrained that paths of these individuals must intersect at least on one point due to perform crossover operation between these parents. This point should be any cell except initial and goal cells.

D. Genetic Operators

As genetic operators, we use one-point crossover and a specific-mutation:

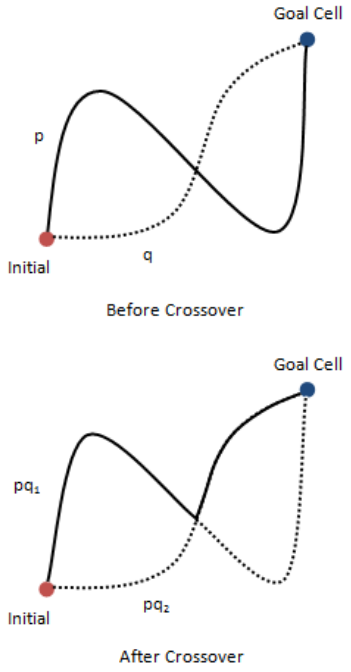

Figure 1. Crossover Operation

1) Crossover: After selecting two parents for mating, in a probability of 80% ($\mu = 0.8$), these parents breed two new individuals with crossover. To perform crossover, parents which have intersecting cells are selected on purpose.

Suppose that p and q are selected as parents. Each path has a set of cells, say $p = \{\sigma_0, ..., \sigma_k\}$ and $q = \{\theta_0, ..., \theta_l\}$. It is absolute that $\sigma_0 = \theta_0 = c_{1,1}$ and $\sigma_k = \theta_l = c_{n,n}$. Now assume that $\sigma_i = \theta_j$ where $1 \leq i < k$ and $1 \leq j < l$ is selected as a crossover point. If so, new individuals' paths are generated as follows:

$$pq_1 = \{\sigma_0, ..., \sigma_{i-1}, \theta_j, ..., \theta_l\}$$

$$pq_2 = \{\theta_0, ..., \theta_{j-1}, \sigma_i, ..., \sigma_k\}$$

Put it in a nutshell, parents switch their sub-paths to perform crossover and generate new offsprings. One of the offsprings takes the sub-path from initial cell to selected intersection cell (excluded) of *parent one* and concatenates with the sub-path which is from selected intersection cell (included) to goal cell of *parent* two. The other offspring takes the sub-path from initial cell to selected intersection cell (excluded) of *parent two* and concatenates with the sub-path which is from selected intersection cell (included) to goal cell of *parent one*. The illustration is given in Figure1.

2) Mutation: In mutation phase, new offsprings are mutated in a probability of 5% ($\gamma = 0.05$). Note that each offspring are mutated independently. To mutate an individual's path, we simply select a random cell over the path -except initial and goal cells- and re-generate a new path between selected and final cells randomly. Re-generation of a random path works same with generation of initial population individuals' paths. This operation avoids genetic drift that different new paths to goal cell can be found simultaneously. Illustration of this specific mutation is given in Figure 2. Dashed path shows the new-generated path after mutation.

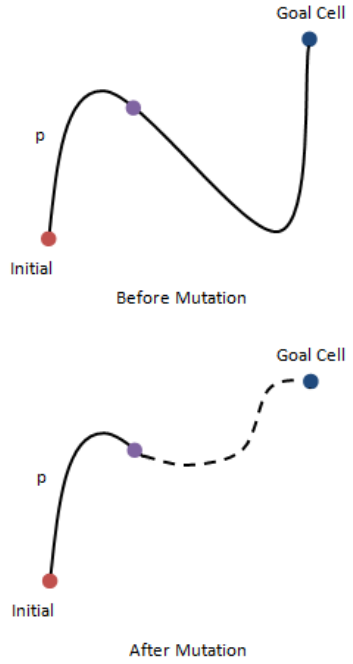

Figure 2. Mutation Operation

E. Selection for Survival

In our approach, every offspring / parent (if crossover operation does not get in action) is added to new population until a fixed number $m - e$ is reached. Here, the e is number of elitists. Just before selection for mating, e best individuals -according to their fitness values- are chosen and put into new population. This method, elitists replacement prevents loss of good solution candidates at next iteration.

V. Experimental Results

As we explain and give details about proposed MOEA, we can give experimental conditions and results of this algorithm. We specify our predefined constraints for our problem domain. We take the number of elitists as 5 (e) and rest of the population size as 250. So, our total population size m is 255 and this size is stable during the evolution process. We take crossover and mutation ratios as same with algorithm definition, $\mu = 0.8$ and $\gamma = 0.05$, respectively. Our grid size n, takes different values; 10,20 and 30. Finally, we evolve the population in 3500 iterations. Note that this maximum iteration number is dependent with n and m. Thus, smaller values of these numbers might cause the algorithm to yield optimal solution in earlier iterations.

Below, we give some outputs through the algorithm. In figure 3 initial population's best individual can be seen. Note that this individual is generated randomly. "X" values represent obstacles, numbers on the cells represent the following path for this individual. For instance, initial population's best individual starts from 0^{th}cell and finishes its path on 97^{th}cell. Notice that some of the numbers can not be seen on the path. That shows overlapped cells on the path exist, or there exist cells which are visited more than one, in other words. We simply get the first value as an output of a cell.

```
Best Fitness of initial population: 4.8322407326751904E-4
Total fitness of initial population: 0.017344047537711536
------------------------------------------------------------
| X  X  X  X  X  X  X  X  X  X  X  .  X  .  X  X  .  .  97|
| X  X  .  .  .  .  .  X  .  .  X  X  .  X  73 X  X  95 96 X |
| X  X  .  .  .  .  .  X  .  .  X  X  .  72 74 76 X  94 .  X |
| X  X  .  .  X  X  X  X  .  .  X  X  71 X  X  77 X  90 93 X |
| X  X  .  .  .  57 .  62 .  68 69 70 .  X  X  80 82 85 91 X |
| X  X  46 47 52 56 61 65 66 67 X  X  X  X  X  X  86 .  .  X |
| X  X  43 48 54 55 X  X  X  .  X  X  X  X  .  .  .  .  .  X |
| X  X  44 42 .  .  X  X  X  .  .  .  .  X  .  X  X  .  .  X |
| X  X  .  X  41 .  X  X  X  .  .  .  .  .  X  .  .  .  .  X |
| X  X  .  .  40 X  X  X  .  .  .  .  .  .  .  .  .  X  X |
| X  .  .  .  39 .  X  .  X  .  .  X  X  .  X  X  .  .  X  X |
| .  .  26 36 29 30 .  X  X  .  .  .  X  .  X  X  .  .  X  X |
| .  23 .  25 28 31 X  X  X  .  .  .  X  .  X  .  X  .  .  . |
| 21 20 19 .  .  .  .  .  .  X  .  X  .  .  .  .  .  .  X |
| 14 .  18 X  X  X  X  X  X  .  .  .  X  X  X  .  .  .  X  . |
| 11 10 X  X  X  X  X  X  X  .  .  .  X  X  X  .  X  .  .  . |
| X  .  9  8  .  .  .  .  X  .  .  .  X  X  X  .  .  X  .  . |
| X  .  6  7  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  X |
| X  .  5  X  X  .  X  X  X  X  .  X  .  X  X  .  .  X  .  . |
| 0  1  2  3  .  .  X  X  X  X  .  X  .  X  X  X  .  .  .  X |
------------------------------------------------------------
```

Figure 3. Initial Population's Best Individual

In figure 4, we show 4^{th}iteration's best individual. Fitness value of whole population and best individual is increased. Also number of visited and overlapped cells are decreased to nearly half with respect to initial population.

Figure 5 gives the 42^{th} iteration of evolution process. Many of the overlapped cells are removed and path is converging to optimal path. This fact can be shown from best and total fitness values of population.

Finally, figure 6 shows the final evolved individual's path. This individual is obtained in 1524 iteration. Note that this path does not cover any overlapped cells and it minimizes both average weight and number of visited cells.

```
Best fitness of #4 th population: 0.0014757807382831671
Total fitness of #4 th population: 0.09071284635429698
# of visited cells in #4th population: 54
```

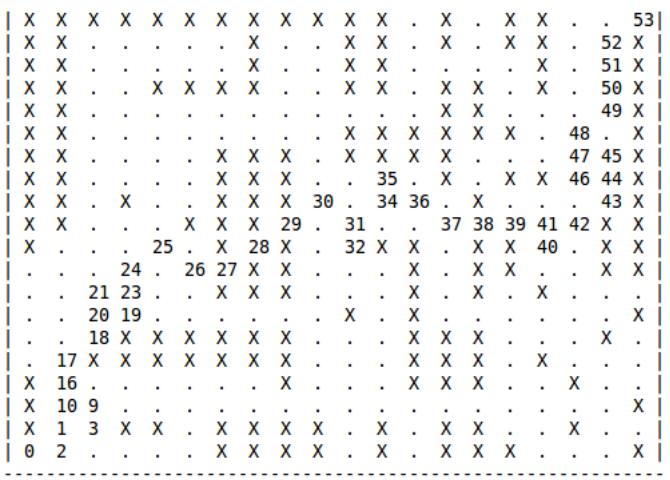

Figure 4. $\#4^{th}$Population's Best Individual

```
Best fitness of #42 th population: 0.0069148860599343555
Total fitness of #42 th population: 1.5019084365655764
# of visited cells in #42th population: 32
```

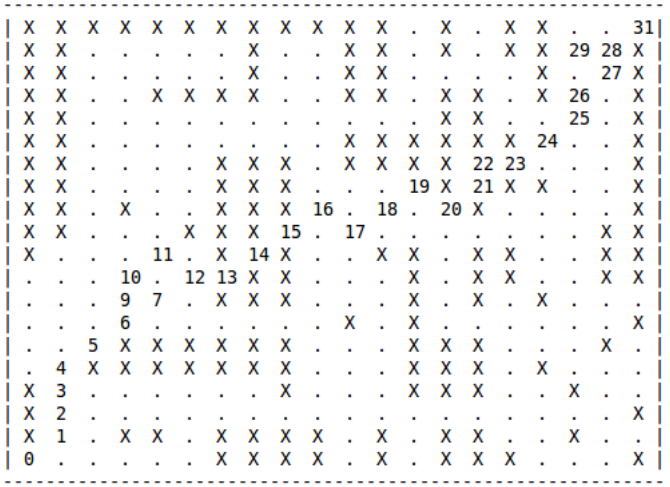

Figure 5. $\#42^{th}$ Population's Best Individual

```
Best fitness of #1524 th population: 0.014083839172489645
Total fitness of #1524 th population: 3.282108005093578
# of visited cells in #1524th population: 26
```

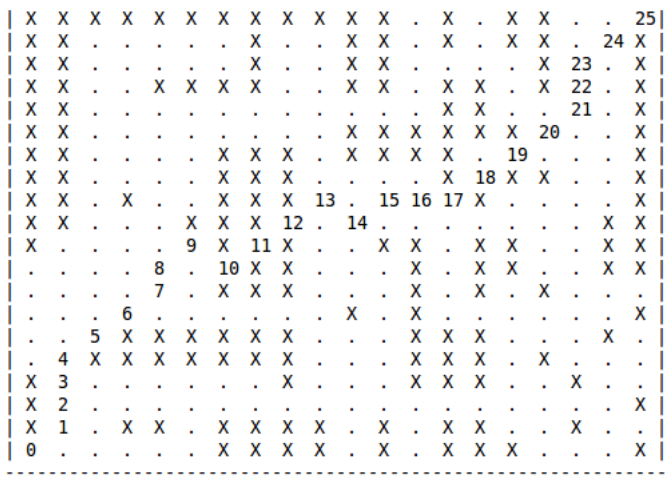

Figure 6. Last Population's Best Individual

VI. Conclusion & Future Work

This paper gives essential aspects about multi-optimization path planning and propose a solution for this planning problem. Results show that evolutionary approaches can be used to determine optimal paths instead of other heuristic methods like A* or D*. In this paper, we consider the problem domain as two-constrained environment. More than two constraints can be considered with just adapting the fitness function of this algorithm.

For future studies, we are planning to extend this algorithm and consider moving targets and obstacles. In other words, we adjust this algorithm to dynamic environments and use the power of evolutionary methods in this manner.

References

[1] Nabeel Bukhari, Ayesha Khan, Abdul Rauf Baig, and Kashif Zafar. Optimization of route planning using simulated ant agent system. *International Journal of Computer Applications*, 4(8):1–4, August 2010. Published By Foundation of Computer Science.

[2] John F. Canny. *The complexity of robot motion planning*. MIT Press, Cambridge, MA, USA, 1988.

[3] Oscar Castillo, Leonardo Trujillo, and Patricia Melin. Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots. *Soft Comput.*, 11:269–279, October 2006.

[4] Carlos A. Coello. An updated survey of ga-based multiobjective optimization techniques. *ACM Comput. Surv.*, 32:109–143, June 2000.

[5] Kalyanmoy Deb and Deb Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. J.Wiley & Sons, 1 edition, June 2001.

[6] G. Dozier, S. McCullough, A. Homaifar, E. Tunstel, and L. Moore. Multiobjective evolutionary path planning via fuzzy tournament selection. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 684 –689, May 1998.

[7] Sven Koenig and Maxim Likhachev. D* lite. In *Eighteenth national conference on Artificial intelligence*, pages 476–483, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[8] Amin Zargar Nasrollahy and Hamid Haj Seyyed Javadi. Using particle swarm optimization for robot path planning in dynamic environments with moving obstacles and target. *Computer Modeling and Simulation, UKSIM European Symposium on*, 0:60–65, 2009.

[9] J.M.A. Pangilinan and G.K. Janssens. Evolutionary algorithms for the multiobjective shortest path problem.