UNIVERSITY OF AGDER

Groupwise Evacuation With Genetic Algorithms

Leonard Loland and Bjørnar Hansen

Supervisors
Morten Goodwin and Ole-Christoffer Granmo

This Master's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

University of Agder
Faculty of Engineering and Science
Department of ICT, 2013

Abstract

In a crisis situation on board a ship, it can be of the utmost importance to have the passengers safely evacuate to the lifeboats in an efficient manner. Existing methods such as marked escape routes, maps and so on are not optimal as pre-planned escape routes may become heavily congested by passengers. The closest lifeboat is not always feasible as lifeboat capacity can be exceeded. Considering that some evacuees are strongly affiliated and would like to evacuate together as a group, it all becomes a very difficult problem to solve. Sub-problems have been modelled, but no existing model combines all of these aspects into account.

We proceed by modelling the area to be evacuated as a time-expanded graph, assuming that future development in hazard severity is known in the form of a survivability percentage for each node. Then we apply a multi-objective genetic algorithm with five different fitness functions that use heuristics to maximize overall survivability and reduce the total egress time if possible. A method has been developed to pick the best evacuation plan out of the pool of potential solutions returned by the genetic algorithm. The solution is compared with Dijkstra's algorithm and randomly generated paths.

Experiments are conducted using these algorithms for both predefined and randomly generated graphs using different parameters. In the tested random graph, the genetic algorithm gives on average 24% better survivability and 3 times better grouping Random algorithms. A fixed network with a known solution was solved 100%.

This genetic algorithm can be used to generate better routing plans that utilizes multiple evacuation routes and lifeboats while taking into account groups, resulting in smoother evacuations which can save more lives.

Contents

List of Figures

Chapter 1

Introduction

In an ongoing crisis situation, on ships and elsewhere, many challenges have to be faced during evacuation. In the case of fire, as it spreads over time it produces an ever-increasing amount of lethal heat and smoke, rendering rooms and corridors hazardous or unusable for evacuation. Emergency response teams can be late and may not have the capacity to assist everyone efficiently, and people are initially left to themselves. The closest emergency exits may become heavily congested as masses of people converge on them, while the nearest lifeboats quickly reach their maximum capacity, forcing other evacuees to make detours to search for alternatives. Escape routes can be rendered too dangerous or unusable, and alternate, perhaps non-obvious, routes have to be used. On top of it all, the information required to make the best course of action, such as the locations of people and hazards, may not be available during the crisis. All this may lead to valuable time and resources being wasted.

While traditional static signs are meant to guide evacuees safely towards exits, they have shortcomings. They do not change if the evacuation routes they indicate become blocked or hazardous. In addition, if too many evacuees decide to take the same indicated nearest escape routes, it can lead to congestion along the route as well as overcrowded lifeboats at the end of the individual routes.

1.1 Background

To mitigate the problems that arise during crisis evacuation, research is being conducted on how personal electronic devices—such as smartphones—equipped with sensors can be applied for management of such situations [1, 12]. Their built-in sensors and communication technologies can both gather information and share it among devices [9], and the aggregate of this information can contribute to clarify the current situation.

By leveraging this kind of real-time information, an automatic evacuation planning system can help resolve some of the challenges faced during a crisis situation, namely how to avoid casualties, congestion, and confusion. It can automatically determine escape routes for everyone present, taking care to lead evacuees away from hazardous situations. It can avoid congestion by taking into account all passengers and their respective escape routes, and it can make sure that evacuees are lead to lifeboats where there are enough room for them.

Earlier work on evacuation planning mainly focuses on off-line solutions. One of the main lines of research focus on mathematical modelling and solutions based on finding maximal flow in networks. A common way of solving the maximal flow problem is linear programming, which was proposed in [4]. Research in this field was reviewed in the survey [6]. Other solutions base themselves off of Dijkstra's algorithm for finding the shortest paths in graphs, as done in [10]. The main drawback of much of the research, including the work in [6, 10], is the fact that it is often considered adequate to focus only on minimizing time expended during evacuation. This neglects the fact that some alternative routes may actually be safer even though they increase the egress time. Minimizing egress time may give satisfactory solutions in many situations where evacuation can be performed before the imminent danger becomes a threat, such as when a hurricane is forecasted to hit shore. In other cases, such as fires, where the danger is much more immediate, it can be important to be able to decide to take the longer route if it has a probability of being safer.

The evacuation research tends to either take the macroscopic approach or the microscopic approach. In [10] the macroscopic approach is considered. Here, the authors look at evacuation during major natural disasters like hurricanes. This sort of evacuation takes place along highways and roads, and focuses on relocating the population closest to the predicted point of impact first. The mode of transportation in this case is by motorized vehicles, and individual persons are considered only as statistical figures of population density and a basis for network flows.

The other approach, which includes the work presented in this thesis, is the microscopic one, where individuals are considered. An example of this is in [5], where evacuation takes place inside constructions such as ships or buildings and the type of hazard is somewhat immediate such as a fire.

Whereas the resolution of the macroscopic approach is coarse enough that most small-scale behavioral effects can be neglected, in the microscopic approach such considerations can provide more accuracy to the model. One such effect is group affiliation: the theory that most people are connected and emotionally attached to certain familiar persons. Taking this into account in the model design, something which is seldom done, should lend credibility to the proposed solution.

1.2 Literature Review

1.2.1 Evacuation Networks

According to [14], the evacuation process has 5 stages. The initial stage happens when an alert is raised. Second, the persons present must react to the alert, and third, a decision must be made to evacuate. The actual evacuation happens in the fourth stage, after which the fifth and final stage is to try to verify that everyone has made it to safety.

For purposes of modelling evacuation networks, [14] suggests the use of 3 steps of representation, analysis, and synthesis. In the first step, representation, the area of interest must be formally modeled as a graph where the nodes represent rooms or more generally areas people can occupy and the edges model the connections between them. There are 3 different types of nodes. First, there are the source nodes, nodes in which people are present at the time of evacuation. Second, the sink nodes are the safe nodes, destinations to which evacuees must go to successfully evacuate. Third are all the other nodes, that is, nodes that are neither sinks nor sources but represent the intermediate nodes which evacuees can enter and exit while evacuating. Nodes and edges are further specified through variables such as length, the number of occupants, and capacity.

The second modelling step is the analysis. Multiple criteria are available for analysing the network representation and the safety of an evacuation plan, of which two are emphasized: Minimum total distance travelled and minimum total evacuation time. The criteria are objectives which shall be either maximized or minimized to increase evacuation performance.

Overall safety is divided into three main categories:

- Minimize egress time
- Minimize routing complexity
- Maximize path reliability

Minimize total evacuation time is included under the egress time category, along with congestion minimization. The other main criterion highlighted, minimize total distance travelled, belongs in the routing complexity category, where the minimization of the longest path, the number of turns, and the number of staircase or ramp traversals is also included. The last category, maximize path reliability, include minimizing the failure of safe areas to accept evacuees, and minimizing edge failures.

The last step of modelling evacuation networks is the creation of an effective evacuation solution. Based on the results of the analysis, evacuation can be improved in three ways. The first is to redesign the network topology itself, increasing evacuation performance by making good design decisions. The second way is to modify the parameters of the existing network, for instance by increasing capacity of nodes or edges. The third way to improve evacuation performance is to modify the evacuation plans, i.e. the paths used by the persons during evacuation.

1.2.2 Group Behavior in Crisis Situations

Groupwise evacuation is grounded in recent social theory. According to the "social attachment" model of human behavior during crisis situations [11], in threatening situations people tend to seek affiliation with familiar persons or attachment figures. This behavior delays the evacuation process; in fact, it has been shown to cause the loss of human lives because people linger together with their group or search for attachment figures instead of promptly evacuating. Evacuation planning without taking into account the strong force of group affiliation would be nigh on pointless, as it is unlikely that evacuees would follow a plan that required group members to go separate ways.

Furthermore, the social attachment model goes against earlier mass panic theories, which claim that chaotic human behavior is the norm when disaster strikes [3]. In contrast those earlier theories, the social attachment model describes evacuation as orderly in most cases. This certainly indicates a higher probability of evacuees displaying an ability to follow the dynamically planned routes than if they were panicking and behaving irrationally.

1.2.3 Ant Colony Optimization

Interesting work has been done in the field concerning stochastic methods for planning safe escape routes. In [5] it is reported that the application of stochastic optimization, here in the shape of the swarm intelligence technique named ant colony optimization (ACO), is a feasible approach to automated evacuation planning, by successfully generating near-optimal escape routes.

[5] is part of a larger project dealing with employing mobile sensing and communication technologies for the betterment of performance in emergencies. The research project aims to create an evacuation planning system, consisting of three parts:

Information collection: Locate people and hazards,
Evacuation route planning: Find optimal paths for safe evacuation, and
Communicate plans: To evacuees and rescue emergency response teams.

ACO lends itself readily to solving the problem of evacuation planning, because it was originally developed for pathfinding in graphs. The area under consideration for evacuation can be modelled as a graph, where evacuees are to move from some source nodes to sink nodes. The graph can then be processed using ACO as follows. Ants are placed in the nodes occupied by people. They then perform weighted random walks through the graph. When an ant reaches a sink, it deposits a set amount of pheromones along its traversed path. The weighing for the random walk is influenced by the pheromone deposits, where more pheromones increase the probability of selecting a given node. Furthermore, an evaporation rate can be added so that pheromone deposits decay over time, ensuring freshness of the path.

In this paper, survivability, path length and congestion are optimized by the algorithm. Path length optimization is basically the goal for the standard ACO; after having located a destination, the ants converge towards the shortest path because they are attracted toward the strongest pheromone trail. To add optimization towards survivability, the pheromone depositing is modified to deposit more on safe nodes and less on hazardous ones. Congestion is taken into account in two ways. First, edges without leftover capacity are not considered when an ant selects where to go. Second, pheromone depositing is influenced by the amount of congestion, decreasing the amount deposited as the used capacity get closer to max.

In summary ACO performs well in the tested scenarios, suggesting that this and other stochastic optimization methods may be good choices for solving the evacuation planning problem.

1.2.4 Evacuation Planning With Multiobjective Genetic Algorithms

Genetic algorithms have been used within the field of evacuation previously.

In [13], the evacuation planning process is described as a three-part process which is performed as a preparatory measure for the case where actual evacuation is needed: Selecting safe areas, finding optimal paths from buildings to safe areas, and selecting the best safe area for each building is included in planning. The first step, selecting safe areas, was done manually. Next, the optimal paths from buildings to safe areas were determined according to safety and traffic. The last step, selecting the best evacuation routes for each building, is then solved with a genetic algorithm.

Two fitness functions were used in the genetic algorithm: distance between building and safe area, and the capacity of safe areas. The multi-objective genetic algorithm NSGA-II was chosen based on its predecessor's promising performance characteristics as documented in [15]. However, some deficiencies of NSGA were noted, particularly:

- The algorithm's complexity at $O(mN^3)$,
- the lack of elitism, and
- the fact that a parameter value had to be provided.

These issues were addressed in NSGA-II, by trading an increase in memory complexity for better computational complexity, adding an elitism mechanism and getting rid of the extraneous parameter.

1.2.5 Shelter allocation using genetic algorithms

Kongsomsaksakul et al. [8] also consider pre-disaster evacuation planning. In their model, the problem is formulated as a Stackelberg game, where the leader is the evacuation planning authority designating shelter locations. The follower is the collection of evacuees, who according to the given shelter locations determine which shelter to move to and by which path.

The GA is employed by the planning authority to place shelters. Given a potential solution from the leader, the evacuees decisions are calculated. The result is fed into the GA's fitness function, which is a weighted sum of constraints on egress time, congestion, and shelter capacity.

1.3 Problem Statement

Fast, efficient and safe evacuation is important during crisis situations. Whereas current approaches to evacuation planning include pre-planned routes, a benefit could be had from providing real-time evacuation planning. Pre-evacuation planning is limited in that it cannot take into account the particularities of crisis situations as they happen; consequently, evacuation operation can be inefficient and unnecessarily dangerous.

Furthermore, it is of interest to take more of human behavior into account than has been done in related work. Specifically, group affiliation is an important aspect of human life, and it affects the evacuation process.

To advance research in a direction which has not seen too much activity, in general stochastic optimization and specifically genetic algorithms should be used for the evacuation planning process.

The crisis situation can be either static or dynamic. In the static case the hazards location and severities do not change, whereas the dynamic case introduces some temporal variations in location and severity.

An evacuation plan's success is measured by how large a percentage of the evacuees are saved. The factor which directly influence this is the amount of time spent in hazardous areas. Consequently it is desirable to minimize this factor, which can be done in multiple ways. Minimization of egress time reduces overall time spent, which implies that it may also decrease the time spent in hazardous areas. Alternatively, explicit routing around these areas avoids the danger altogether. Egress time can be reduced by routing through the shortest available path or by minimizing congestion. Ensuring that evacuees arrive at a lifeboat with room to spare is crucial to achieve a successful and safe evacuation.

1.4 Assumptions and Limitations

We assume that all necessary information about the evacuation scenario is known. This includes the physical layout of the area, where lifeboats are located, where people are located and to whom they are affiliated, hazard severities for each room, and capacities for each room and passages between rooms.

We also assume that once a path has been assigned to a person or group, people will follow that path. This assumption may not be realistic. However, one can imagine that if the assigned paths are considered reasonable by the groups they have been assigned to, it is more likely that they will stick to the path. What, then, makes a path reasonable? Some parameters are obvious. If the route leads the evacuating group toward perceived danger, such as smoke or alarming sounds, they are likely to turn around. Similarly, if the path loops around and leads the evacuees to cross their own path, they will quickly lose faith. Another issue may arise if two or more groups cross paths or otherwise make contact and they are not moving in the same general direction. It is hard to say what would happen in this situation, but it seems likely that the groups could influence each others' behavior. Nevertheless, if appropriate instructions are given beforehand, and if the automatic planning system is thoroughly verified and trust-inspiring, evacuees should be more likely to stick to their routes.

1.5 Contribution

The evacuation planning problem can be seen as an optimization problem, where the target is to find the optimal evacuation routes in a given scenario. As a novel approach toward solving the problem, we employ genetic algorithms as the optimization mechanism.

By improving the process of evacuation through providing escape routes adapted to current situations and conditions, it is our hope that we can contribute to a more efficient and safer evacuation process.

Our research plows new ground by applying genetic algorithms to generate optimal paths for use in an evacuation scenario. While research exists which apply genetic algorithms within the field of evacuation planning, most if not all use them for other tasks than path generation. Furthermore, we add complexity to the problem by considering some behavioral issues with groups as a fundamental social structure.

In this thesis, we present an adaption of genetic algorithms which handle pathfinding in graphs. Specifically, the chromosome design and the genetic operators are adapted towards the nature of the path-finding problem. Of important note is that the genetic algorithms are multi-objective in nature, as multiple criteria are important in the design of evacuation routes. Finally, we present a comparison of performance between algorithms in several different simulated evacuation scenarios.

The structures that are to be evacuated are modelled as graphs with uni-directional edges. Edges and nodes have certain capacities, which determine how many people can be present before congestion kicks in. Nodes can have various degrees of hazard severity which, depending on the complexity of the simulation, may change as time passes.

1.6 Outline

In Chapter 2 we describe related work that has been done in the fields of evacuation planning and genetic algorithms. In Chapter 3 we describe the specifics of the genetic algorithms used. Following that is a discussion and comparison of the results in Chapter 4, and lastly the paper concludes with Chapter 5 where suggested future work is included.

Chapter 2

Theory

2.1 Evacuation Routing

Several terms used throughout this thesis describe concepts related to evacuation. The following definitions are some of the most important words.

Designated safe areas which evacuees are trying to reach have several names which mean approximately the same. They may be called fire exits (or simply exits), endpoints, shelters, safe rooms, safe nodes, or even lifeboat. The most important distinguishing feature is whether the safe area has limited capacity (in the case of lifeboats or other spatially limited locations) or if it is of virtually unlimited capacity (such as an exit leading to an unbounded area outside of a building).

The terms egress time is the time it takes from the start of evacuation until all evacuees have relocated to the safe areas.

Evacuation network describes a graph model of the building being evacuated. Rooms are nodes, connected by edges to their neighboring rooms.

Congestion is the issue where more people are passing through an area than the area can take, leading to blocking, crowding, slowdowns and potentially dangerous situations.

Endpoint capacity describes the limited capacity some safe nodes have, such as lifeboats.

Capacity relates to the amount of people a room can hold before congestion starts occurring.

2.2 Graph Theory

From discrete mathematics, a graph G is a representation of objects that are related with each other. Interrelated objects are represented using a set of nodes N and a set of edges E, where pairs of nodes are related using one or more edges.

In a directed graph, edges have direction. A directed edge (m, n) is a one-way relation from node m to node n where $m, n \in N$ and $(m, n) \in E$. It is an independent edge from the reversed edge (n, m) which may or may not exist in E.

Edges can be assigned values. The length value of an edge could represent the metric distance from the center of one node to the center of another node.

An example graph is given in Figure 2.1. The graph is drawn with circles and arrows, representing nodes and edges, respectively. The nodes are labeled 1, 2, 3 and 4, while the edges have been left unlabeled to avoid cluttering the figure.

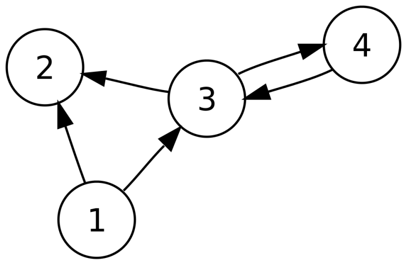


Figure 2.1: A simple graph representation.

2.2.1 Time-Expansion

Time-expansion is a common technique to make static models dynamic. For instance, if a graph represents some data at some specific time step, it can be remodelled to represent data that varies over multiple time steps. The original nodes are copied once more for each additional time step. Edges that previously connected nodes within the same time step now connect nodes across time steps.

It is also possible to connect nodes across multiple time steps. If it takes two time steps to move from any node m to any other node n in the original static network, then there is an edge from node $m(t)$ to node $n(t+2)$ for each time step t in the expanded network. This allows traversal from n to m at any time step t.

After a time-expansion, holdover edges are used to connect a node to the next time copy of itself $x(t+1)$. In an evacuation, moving from node $x(t)$ to $x(t+1)$ is a equivalent of waiting in x one time step.

2.2.2 Network Flow

In graph theory, a graph is called a network when edges are associated with flow values. Flow is the rate of which a quantity moves from one node to another. These nodes are called source and sink, respectively.

Maximum flow is the amount of flow that be put through from a single source to a single sink using multiple paths and without to exceeding any single capacity. Maximum flow is equivalent to maximum throughput. Consider a network connected from x to y to z. If capacity $c((x,y)) = 1$ and capacity $c((y,z)) = 2$, then the maximum throughput is 1 and not the maximum flow of an individual edge which is 2.

Maximum flow between multiple sources S and multiple sinks T can be modelled using a workaround [4]. First, two additional nodes are added to the original network called super source s_0 and super sink t_0. Then, directed edges connect the super source to each source and each sink to the super sink so that

- $\forall_{s \in S} \left[(s_0, s) \in E \wedge (s, s_0) \notin E\right]$, and
- $\forall_{t \in T} \left[(t, t_0) \in E \wedge (t_0, t) \notin E\right]$.

These super edges have infinite capacity so that the maximum flow is only constricted by the capacities in the original network.

Congestion during evacuation has been modelled using the concept of flow [6]. In this case, a flow value f is the number of evacuees that moves from one node to another per time unit. This flow is bounded by a capacity c so that $f \in [0, c]$.

2.3 Genetic Algorithms

Genetic algorithms (GAs) use the model of natural evolution as an optimization framework, incorporating survival-of-the-fittest, mating and reproduction.

The algorithm starts by creating a population of random solutions which also are called chromosomes. These chromosomes are evaluated using a fitness or objective function which measures the performance of a solution. Solutions with higher performance have a better chance of being selected for breeding, where potential solutions are combined according to predefined rules of crossover. Following this, each solution has a probability of going through a random mutation process, and finally the population of old solutions is replaced by the newly generated ones, and the process is repeated until a stopping condition is met.

This process is presented in Algorithm 1, and it is further detailed in the following subsections.

Algorithm 1 Genetic Algorithm

```
1:  function GENETIC ALGORITHM
2:      $P \leftarrow$ randomly generated population
3:      repeat
4:          EVALUATEFITNESS($P$)
5:          $Q \leftarrow \varnothing$
6:          while $|Q| < |P|$ do
7:              $a, b \leftarrow$ SELECTION($P$)
8:              $c, d \leftarrow$ CROSSOVER($a, b$)
9:              $Q \leftarrow Q \cup \{c, d\}$
10:         end while
11:         MUTATE($Q$)
12:         $P \leftarrow Q$
13:     until desired end condition is met
14: end function
```

2.3.1 Fitness Function

The fitness function represents the task for which optimization is desired. By evaluating this function, a measure of the performance of a solution is acquired. Fitness functions can often be severely expensive, dependent on the problem at hand. This is one of the main drawbacks of GAs, although the rapid increase in computational power experienced throughout the last decades has somewhat mitigated this. On the other hand, it is possible to use less complex approximations of the real objective function to avoid the computational cost, at the cost of decreased accuracy.

2.3.2 Chromosomes

The chromosome is the most central concept in GA. Encoded as data points (alleles) in the chromosomes are the parameters which the fitness function evaluates.

Chromosome encoding can be whatever is convenient and fits the problem domain: a binary string, list of integers, enumeration of possible fitness function parameters, and so on. The implementation of crossover is dependent on chromosome encoding.

2.3.3 Crossover

Crossover is genetic recombination, between two or, in some cases, more chromosomes. The simplest forms of crossover between two chromosomes $c1$ and $c2$ is the one-point crossover and the uniform crossover.

In one-point crossover, a random position p is selected. Then two new chromosomes are formed: the first consisting of the part of $c1$ from its start up till p and the part of $c2$ from p till its end. The second child chromosome is generated likewise with $c1$ and $c2$ swapped.

For uniform crossover, each allele in the children has a 0.5 probability of coming from either parent. Each allele not included in the first child will be included in the second child and vice versa.

2.3.4 Selection

Several different selection mechanisms exists. After the fitness function has been used to rank all the solutions in the population, this is used to choose chromosomes to be fused for crossover.

Tournament selection is easy to implement and often used. In this selection method, two or more solutions are randomly picked from the pool. Then their fitness is compared, and the one with the best fitness is the winner of the tournament and is the chosen one.

Fitness proportionate selection, which is also know as roulette wheel selection, is a method where solutions are given a selection probability directly proportional to their fitness, so that the most fit solution has the highest chance of being selected.

2.3.5 Mutation

Mutation introduces some random variation into the gene pool. With a small probability, each allele can be swapped for a new, randomly generated value.

Mutation can somewhat counteract the problem of genetic drift and fixation by randomly inserting alleles in the population.

Genetic drift occurs when offspring is randomly generated. The frequency of the occurrence of a particular allele will fluctuate from generation to generation, but it is dependent on the allele's occurrence in the previous generation: a high frequency in the parent generation leads to a high probability of a high frequency in the offspring. Due to this effect it is likely that after a number of generations one particular allele will be very dominant in the population, or even that other variations are lost and only one allele occurs in the population. This is called genetic fixation.

2.4 Multi-Objective Optimization

In many optimization problems, including the problem of safe evacuation as defined earlier, there is more than one objective. And the fact that these objectives often conflict further complicates the matter. This motivates the search for techniques, called multi-objective optimization (MOO), which facilitates the search for optimal solutions while taking into account all the defined objectives.

Extensive research has been conducted on using GAs for MOO; for an introduction, see [7]. The main differences in these approaches compared to standard GAs are modified fitness functions, and techniques that promote and preserve diversity in the GA's population. Of particular note are the methods that go under the Pareto ranking umbrella.

In the Pareto ranking class of methods the idea of Pareto optimality is used to rank potential solutions. Pareto ranking is used to avoid having to decide whether one objective is more important than another. Solutions that receive the same Pareto rank are equal in fitness. Other methods are then used to select between solutions with the same rank.

2.4.1 Traditional GA for MOO

Traditional GA can be used, even when there are multiple objective functions. This is accomplished by combining the values obtained from the objective functions. Several combination methods can be used.

The simple summing approach combines the fitness values by summing them. It is given by $z = \sum_{i \in I} z_i$, where I is the set of indices for the different fitness functions and z_i is the normalized fitness value obtained from fitness function i. As further explained in [7], a weight vector can be applied to the fitness values to emphasize some more than others.

A second way to combine fitness values is by calculating the Euclidian norm of the fitness vector. That is, $z = \sqrt{\sum_{i \in I} z_i^2}$.

The last method we use is a prioritized fitness ranking approach where the ranking is determined by the nature of the heuristics employed as objective functions. The objective functions are ranked, where the most specific or important ones receive the highest rankings.

2.4.2 NSGA-II

Originally presented in [2] as an improvement of its predecessor NSGA, the NSGA-II (Non-dominated sorting genetic algorithm version 2) algorithm employs the ideas of Pareto optimality and crowding distance for maintaining diversity in the population. This technique is a modification of the standard GA.

NSGA-II uses Pareto optimality to sort the population and arrange it into Pareto fronts. Within each front, no solution is strictly better than any other solution. Solutions within a front are assigned the rank of that front; e.g. solutions in front 0 get the rank 0 (which is the best rank), front 1 gets rank 1 and so on. An example is shown in Figure 2.2. For purposes of illustration, in this figure only two fitness functions were applied.

Crowding distance is a measure of how close a particular solution is to other solutions in the fitness space. Smaller distances indicate that the solution is located in a crowded region, meaning that there are multiple solutions with comparatively similar fitness. In Figure 2.2, the crowding distance in two dimensions for one solution is shown, marked as cd_1 and cd_2. The total crowding distance is the sum of distances over the fitness space.

For reference, the crowding distance algorithm given in [2] is reproduced in Algorithm 2.

To calculate the crowding distance, for each fitness function the population is sorted according to its value. Then, for each solution, the distance between the two solutions closest to it is added to the crowding distance.

The binary tournament selection mechanism included in NSGA-II does not use fitness directly. Instead, it takes both rank and crowding distance into account. If a solution has a better rank, that solution is selected. Otherwise, if the ranks are

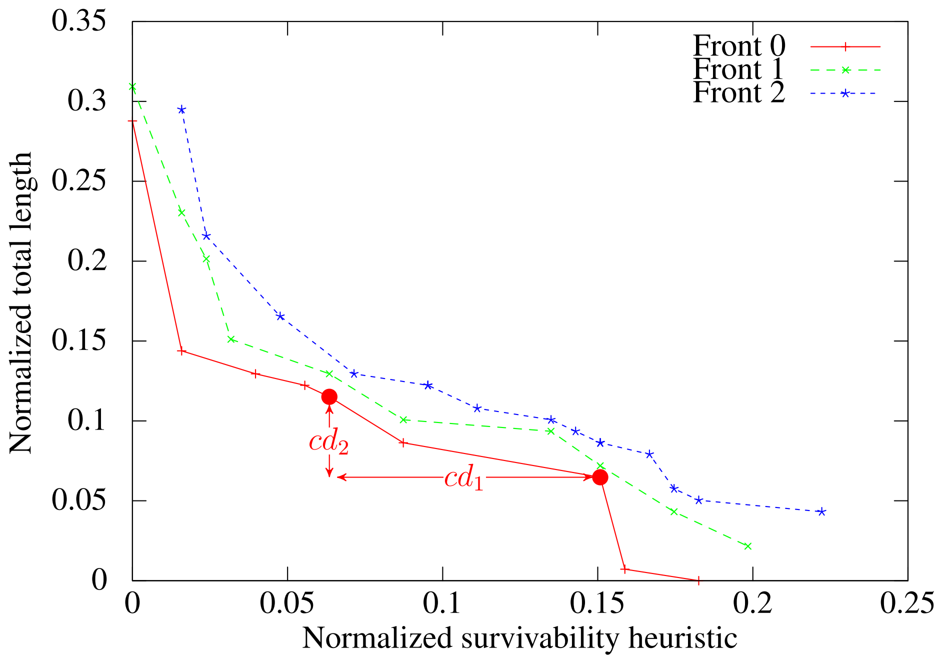


Figure 2.2: Pareto fronts

Algorithm 2 Crowding distance calculation

```
1: function CrowdingDistanceAssignment(P)
2:     $l \leftarrow |\mathcal{I}|$
3:     for each $i$ do
4:         $\mathcal{I}[i]_{\text{distance}} = 0$
5:     end for
6:     for each objective $m$ do
7:         Sort$(I, m)$
8:         $\mathcal{I}[0]_{distance} = \mathcal{I}[l]_{distance} = \infty$
9:         for $i = 2$ to $(l - 1)$ do
10:            $\mathcal{I}[i]_{distance} = \mathcal{I}[i]_{distance} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)/(f_m^{\text{max}} - f_m^{\text{min}})$
11:        end for
12:    end for
13: end function
```

equal, but the crowding distances are different, then the solution with the largest crowding distance is selected. Ties are broken arbitrarily.

Algorithm 3 can be described as follows. First, the current and previous populations are combined and ranked according to Pareto fronts (lines 3–4). Then, starting from the first front, the fronts are added to a new population until the next front cannot be added as a whole without exceeding the stipulated population size. Crowding distance is calculated for every front that is added (lines 6–10). To complete the new population, individuals are selected from the next front in descending crowding distance order (lines 11–12). Finally, on line 13 the next population is generated as usual through selection, crossover, and mutation.

Algorithm 3 NSGA-II

```
1: function NSGA-II
2:     $R_t \leftarrow P_t \cup Q_t$
3:     $\mathcal{F} = \textsc{FastNondominatedSort}(R_t)$
4:     $P_{t+1} \leftarrow \varnothing$
5:     $i \leftarrow 1$
6:     while $P_{t+1} + |\mathcal{F}_\rangle| <= N$ do
7:         $\textsc{CrowdingDistance}(\mathcal{F}_\rangle)$
8:         $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_\rangle$
9:         $i \leftarrow i + 1$
10:    end while
11:    $\textsc{Sort}(F_i)$
12:    $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_\rangle[1 : (N - |P_{t+1}|)]$
13:    $Q_{t+1} \leftarrow \textsc{NextGeneration}(P_{t+1})$
14:    $t \leftarrow t + 1$
15: end function
```

Elitism is implemented through the combining of the current and previous populations, and selecting the most fit individuals from this composite population. Due to this elitism mechanism the process during the very first generation is different from the rest, seeing as there is no previous population to combine with. Instead, the population is used as-is, and the regular process of selection, crossover, and mutation is applied as usual. Having generated this new population, the two populations required for the continuation of the algorithm are now available.

2.5 Dijkstra’s Algorithm

Dijkstra’s algorithm is a well known algorithm within computer science. It is used to find the minimum cost or distance from one node to another in a graph. The algorithm is described in a way that can be used later for the solution in Chapter 3.

Let graph $G = (N, E, s, t)$ be a set of nodes N interconnected by a set of bidirectional edges E. Without loss of generality, assume that G is not a multigraph, i.e. that it has no parallel edges. Each node $n \in N$ is associated with a given cost $c(n)$. Similarly, each edge $e \in E$ is associated with a given cost $c(e)$. Further, define the length of a path $\mathcal{P} \subset E$ between two nodes as the sum of costs of all nodes and edges included in $\mathcal{P}$. The path with the minimum length from the starting node $s \in N$ to the target node $t \in N$ is resolved by Dijkstra’s algorithm as follows.

The distance from s to all nodes except s are initially set to infinity, so that

$$\forall n \in N, d(n) = \begin{cases} 0 & n = s \\ \infty & \text{otherwise.} \end{cases}$$

Additionally, we define the previous node of n as All nodes in N are then added to a priority queue where the distance determines ordering in ascending order. Thus, the starting node s is placed first in the queue.

While the queue is not empty, the first node m is polled from the queue. If m is the target node t, the target has been reached and the loop is ended Else if $d(m) = \infty$, it means that the target cannot be reached.

Otherwise, for each neighbor node n of m, if $d(m) + c(e) + c(n) < d(n)$, then $d(n) \leftarrow d(m) + c(e) + c(n)$ and $p(n) \leftarrow m$. Then, the next iteration of the loop commences.

The final step of the algorithm is to construct the actual path $\mathcal{P}$. To accomplish this, start by setting the current node n to t. Then, while $p(n) \neq s$, prepend the edge between $p(n)$ and n to $\mathcal{P}$ and set the current node to $p(n)$. Finally, prepend the edge between s and n. Now the path is complete, and can be returned.

By modifying the previous set to store edges rather than nodes, this procedure can be extended to multigraphs as well.

Chapter 3

Solution

3.1 System Architecture

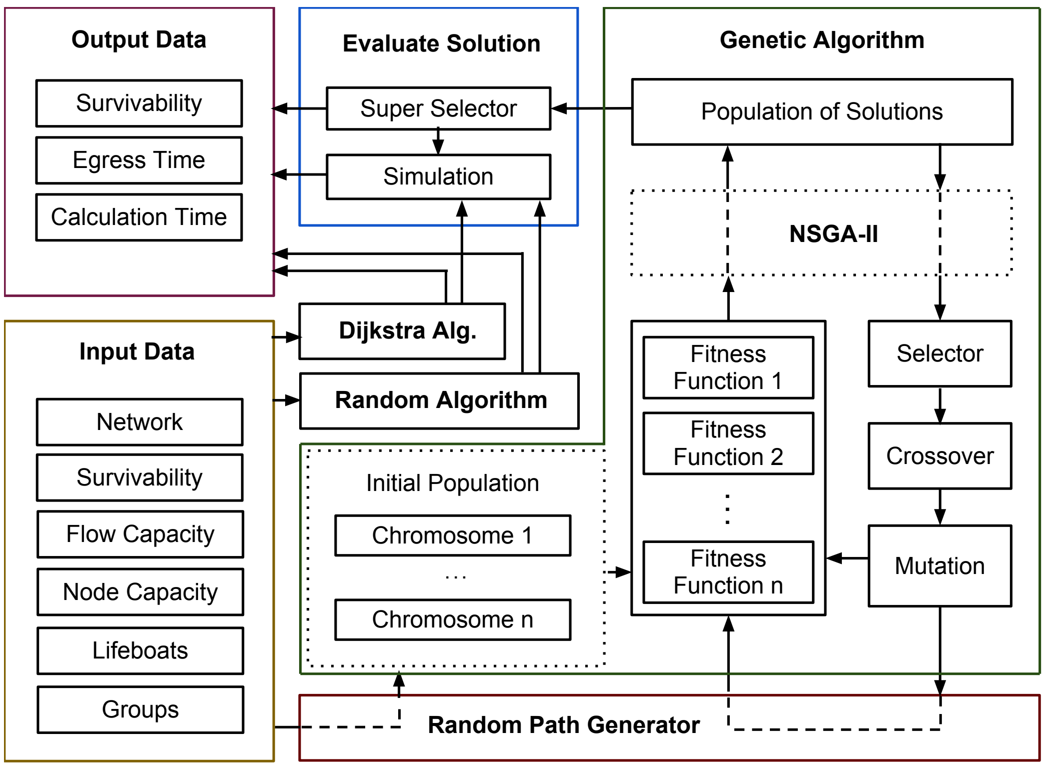


Figure 3.1: The overall system architecture with data flow.

This section will briefly describe the major components found in the overall system architecture, Figure 3.1. Related minor components are included in the major descriptions.

3.1.1 Network Model

As part of the input data, we model an evacuation using a time-expanded network. Nodes and edges have properties that provides evacuation semantics.

A node n

- can be either a room (source), lifeboat (sink), super source or super sink,
- holds zero, one or more evacuees without exceeding its capacity $c\left(n\right)$, and
- has a survivability $\sigma\left(n\right)$ so that $\sigma\left(n\right) \in [0, 1]$ indicating the probability of survival for one timestep.

Evacuees are randomly divided into groups that will, upon evacuation, be gathered and moved towards the same lifeboat. The assigned evacuation route should increase the overall survivability also taking into account future survivability and congestion developments.

This dynamic feature has been modelled using a time-expanded network, we can model hazards that changes over time, and therefore also take into account future developments. Each node can have a different survivability each time step. If used systematically, it can be used to simulate hazard severity change and spread over time.

It is important to note that such dynamic input data be set according to the granularity of the time-expansion, i.e. the size of each time step. For instance, it is more likely to survive one second than one hour when one is potentially exposed to hazards.

An edge e

- has a length $l\left(e\right)$ which is the length from the center of a node to the center of another node,
- has a capacity $c\left(e\right)$ which is the maximum number of evacuees per timestep that can move simultaneously along it,
- has a flow value $f\left(e\right)$ which is the actual number of evacuees per timestep using it so that $f\left(e\right) \in [0, c\left(e\right)]$, and
- has a separate directed edge in the reverse direction with equal properties.

The latter property of an edge is a bidirectional property that is used for lifeboats and super edges. It allows to model evacuees to move one way only by excluding one of the two edges. For lifeboats, it has been used for the assumption that once an evacuee enters a lifeboat, one does not desire to leave. For super edges, it has been used according to theory 2.2.2.

Edge capacity $c(e)$ as suggested in [6] has been extended by node congestion $c(n)$. While the former reduces the flow of which a quantity moves from one node to another, the latter limits the number of people that can fit inside a room. We use the term lifeboat capacity to specifically refer to a lifeboat's node capacity.

3.1.2 Random Path Generator

The random path generator is a utility that generates a random path from a source node to a target node. First, an empty set of current nodes is initialized, and the location node of an evacuee is added to the set. In addition, an empty map from nodes to previous edges is created. Then, while there still are nodes in the set, a random node is selected and removed. Every neighbor of this node is now examined. If a previous edge already exists in the map, the neighbor is skipped; otherwise, the edge between the node and its neighbor is registered as the previous edge for the neighbor. Now, if the neighbor node is the super sink, the traversal is complete and the loop is ended. Otherwise, the iteration continues.

A valid generated path is the sequence of the randomly selected edges that starts at the location of the associated person or group and ends at the super sink.

Due to the semantics of time-expansion, the target node is not guaranteed to be visited. This happens if the time it takes from moving from source to destination exceeds beyond the time step of the destination node. If source at time t=2 and destination at time t=3, but it takes 2 or more time steps to reach destination node at t=3.

3.1.3 Genetic Algorithm

The random path generator is used to create an initial population of solutions that will be further improved by iterations.

Solutions within a population are encoded as chromosomes (3.2).

Our GA implementation uses NSGA-II as described in 2.4.2. Like other GAs, NSGA-II depends on implementation specific components. These components are used every iteration as follows:

- selector selects solutions pairwise until a subset of solutions have been selected,
- crossover combines the pairs of solutions in effort to take the best from both solutions,
- mutation randomly modifies each solution with a certain probability to explore other solutions, and
- five different fitness functions, one for each objective, to distinguish between good and bad solutions.

Iteration stops after some termination criterion is met. In our implementation, iteration stops after completing a fixed number of iterations.

After termination, the best solution must be selected from the population. The population contains solutions that are optimized for one or more of the five different objectives. Some objectives are related and some are more important than others. This problem is handled by our super selector (3.6).

3.2 Encoding of Chromosomes

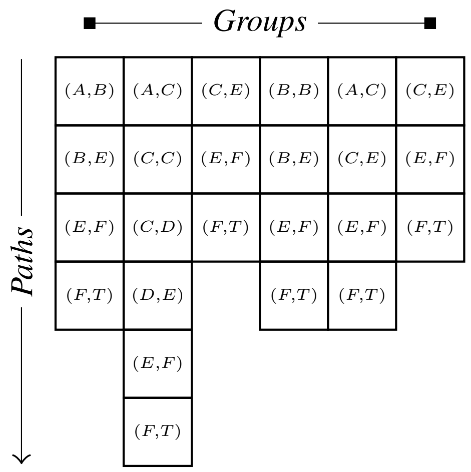


Figure 3.2: A chromosome containing an evacuation routing plan.

A chromosome contains path assignments for each group as shown in Figure 3.2. In this figure, the network includes nodes A-E, sink F and super sink T. The elements of each path, e.g. (A, B), are edges. Each column is a path assignment to either a person or a group which is initially located in the first node in the path. Edges are used in the path to support cases where multiple edges connect two nodes, such as if two oblong, parallel rooms are connected with two or more doors.

However, this issue can be solved differently, which would have reenabled the use of nodes instead of edges. Instead of translating directly from room to node and from door to edge, one could add additional nodes and edges in such a way as to avoid the multiple edges. For the example of two parallel rooms with two connecting doors, this could involve adding two nodes (instead of one) for each room, and connecting them in a rectangular pattern.

Defining paths as a sequence of arcs instead of nodes removes the need to query the network for the arc between each pair of nodes, at least in our implementation.

Paths should obey the following constraints at all times:

- The source of the first arc in the path should be equal to the node where the associated group currently is.
- The path should end at the super sink and not an arbitrary sink.

3.3 Genetic Operation

The GA starts by selecting a pair of parent chromosomes are selected using binary tournament selection as used in [2] and then performing a crossover using these.

3.3.1 Crossover

A multi-point crossover operator for recombining a pair of two-dimensional chromosomes has been immplemented, by using one-point crossover once for each group represented in the chromosome. The crossover point in each parent is randomly selected among potential, valid crossover points. For a crossover point at node n to be considered valid, an edge with a target node n must exist within the corresponding path in both parent chromosomes. However, the common node does not need to be traversed at the same time. An example is given in Figure 3.3.

If the crossover operation creates an invalid child, one of the parent is passed as child instead. Because different lengthed children are created, the path may extend beyond the time-expansion. By ensuring that the initial population is valid and only passing valid solutions as children, this problem will not occur.

It is important to note that this crossover operator can take two identical parents and still produce distinct children. Usually, a chromosome which is recombined

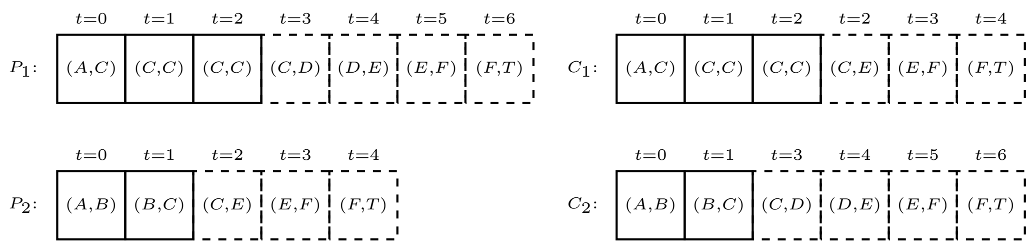

P_1: t=0 (A,C) t=1 (C,C) t=2 (C,C) t=3 (C,D) t=4 (D,E) t=5 (E,F) t=6 (F,T)
P_2: t=0 (A,B) t=1 (B,C) t=2 (C,E) t=3 (E,F) t=4 (F,T)
C_1: t=0 (A,C) t=1 (C,C) t=2 (C,C) t=2 (C,E) t=3 (E,F) t=4 (F,T)
C_2: t=0 (A,B) t=1 (B,C) t=3 (C,D) t=4 (D,E) t=5 (E,F) t=6 (F,T)

Figure 3.3: An example crossover operation.

with itself will produce children that are perfect copies of itself, and hence with no possible improvement. However, due to the way chromosomes are encoded and crossover is implemented, in this case a parent mating with itself has the possibility of producing offspring which are different and may be better than its parent. This effect arises because edges, which are reusable at different time steps, can occur several times in the same chromosome, which can lead to a single chromosome having multiple time-shifted crossover points with itself. Nevertheless, offspring that are identical to their single parent will still occur if the exact same crossover point is used for both of the parents.

Reuse of edges allows for two things. Firstly, it allows waiting in a node by following the holdover edge two or more times in sequence. Secondly, it also occurs when paths are circular. Regardless, such solution paths will be evaluated by fitness functions and handled accordingly.

3.3.2 Repair Function

In a time-expanded network, crossover breaks the sequential timing of the path. Consider again Figure 3.3 above. If a common node C for parents P_1 and P_2 is traversed at $t = 3$ and $t = 2$ respectively, the child paths are broken because the node C from which $(C, E)_{t=2}$ emanates is not the same one occurring in $(C, E)_{t=3}$. The correct edges are traversed, but not at the correct time.

To fix the timing of a path, the repair function defined in listing 4 can be applied after a crossover.

where

- $\mathcal{P}$ is an ordered set of edges defining a path,
- $\mathcal{P}_k$ is the k-th element in $\mathcal{P}$,
- TIME(e) returns the time step of the source node of an edge e,

Algorithm 4 Repair Path Timing

```
 1: function RepairTime($\mathcal{P}$)
 2:     if $|\mathcal{P}| = 0$ then
 3:         return
 4:     end if
 5:     $\mathcal{P}_0 \leftarrow$ ZERO($\mathcal{P}_0$)
 6:     repaired $\leftarrow false$
 7:     while $\neg$ repaired do
 8:         repaired $\leftarrow true$
 9:         for $i = 1$ to $|\mathcal{P}| - 1$ do
10:             $a \leftarrow$ TIME($\mathcal{P}_{i-1}$) $+$ LENGTH($\mathcal{P}_{i-1}$)
11:             $b \leftarrow$ TIME($\mathcal{P}_i$)
12:             if $a < b \wedge$ PREV($\mathcal{P}_i$) $\neq null$ then
13:                 $\mathcal{P}_i \leftarrow$ PREV($\mathcal{P}_i$)
14:                 repaired $\leftarrow false$
15:             else
16:                 if $a > b \wedge$ NEXT($\mathcal{P}_i$) $\neq null$ then
17:                     $\mathcal{P}_i \leftarrow$ NEXT($\mathcal{P}_i$)
18:                     repaired $\leftarrow false$
19:                 end if
20:             end if
21:         end for
22:     end while
23: end function
```

- LENGTH(e) returns the length of an edge e assuming 1 length unit corresponds to 1 time unit,
- ZERO(e) returns the first time instance of an edge e,
- PREV(e) returns the previous time instance of an edge e or $null$ if the previous edge is beyond the network time-expansion, and
- NEXT(e) returns the next time instance of an edge e or $null$ if the next edge is beyond the network time-expansion.

a represents the time step of an edge e + the time it takes to traverse e. a should then be equal to b, the time step of the next edge in the path. If not, replace e by PREV(e) or NEXT(e) to make this true.

3.3.3 Mutation

The one-dimensional mutation mutates each path of each chromosome in the population by a predefined probability. The mutation uses the random path generator to generate a new path that starts from the same origin and ends at the super sink.

3.4 Evacuation Criteria

In this section we tie the evacuation criteria described in Section 1.2.1 to the ones used in the optimization techniques detailed in the next section.

3.4.1 Egress Time

The distance an evacuee has to travel has an impact on the probability of survival in the way that increased distance means more time spent in a potentially hazardous environment. However, it will usually be beneficial to make a detour if it means avoiding hazardous rooms. Moreover, spending more time during evacuation may also influence the level of congestion.

A suitable measure for egress time is the total time spent on evacuation, meaning the sum of time spent for each evacuee. To minimize this value one can start by minimizing congestion occurrences. Congestion can be a big source of wasted time when the paths used for evacuation have relatively low capacity compared to the flow of people passing through.

3.4.2 Congestion

Congestion may occur when the available exits are limited, or when a large number of evacuees pass through a space too small to accommodate them. This causes crowding and thereby delays, which in turn cause people to wait longer in possibly unsafe rooms. It is therefore desirable to keep congestion to a minimum or if possible avoid it completely.

3.4.3 Route Complexity

Route complexity can be approximated by the total distance travelled. Even though complexity also includes the number of path segments, direction changes and so on, the total distance travelled is a complexity measure because shorter paths will have less features, hence being less complex. Total distance travelled is the sum of distance travelled for each evacuee.

3.4.4 Reliability

The reliability of a path indicates the likelihood of the path to lead evacuees to safety. It can be approximated by hazard severity or survivability. Hazard severity represents the probability of survival for an evacuee in the room. It is 1 if there is certainly no hazard, 0 if there definitely is one, or a number in between when the knowledge is uncertain.

Survivability is the inverse of hazard severity. It is a probabilistic estimate of the likelihood of survival in a certain room.

3.4.5 Endpoint Capacity

While endpoints in buildings have unlimited capacity in the sense that once a person has passed through the exit it can be used again, this is not generally the case. In some situations, e.g. on a ship at sea, the lifeboats have limited capacity. Failure to appropriately assign evacuees to endpoints will cause unnecessary delays if the evacuees must go searching for different endpoints.

In situations where the safe areas have limited capacity, reliability is influenced by whether evacuees are able to enter their designated safe area.

3.4.6 Sidebar / sub-problem: Bin-Packing Evacuees Into Lifeboats

The issue of distributing groups of evacuees into limited-capacity lifeboats can be seen as an instance of the bin-packing problem. In this case, the size of each group is considered its volume, and the number of bins – the lifeboats – are limited.

This problem becomes harder as the total number of passengers approaches the total lifeboat capacity, or as group sizes increase, because the number of possible solutions decrease. However, in realistic situations there will be much more room in the lifeboats than there are passengers present.

In any case, this issue can be taken care of by the genetic algorithm, thus requiring no special consideration.

3.5 Fitness Functions

All fitness functions use heuristics that are motivated by the evacuation criteria as given in Section 2.1. The survivability heuristic is a sum of the probabilities that there is a hazard in all nodes along all the given paths. Total distance measures the total length of the evacuation paths. The endpoint capacity heuristic tells of how many evacuees there were no room for in their assigned safe nodes, while the congestion heuristics counts all evacuees that cause nodes or edges to exceed their capacity.

All the heuristics are created in such a way that the lower values they return, the better a solution is assumed to be. This allows for straightforward application of the heuristics as fitness functions in a minimization scenario.

The heuristics are described in more detail in the following subsections and in algorithm listings 5 through 9.

3.5.1 Survivability

Algorithm 5 Survivability Heuristic

```
1: function SurvivabilityHeuristic($C$)
2:     return $\sum_{\mathcal{P} \in C} \left[ \sum_{n \in \mathcal{P}} [1 - \textsc{survivability}(n)] \cdot \textsc{EntitySize}(\mathcal{P}) \right]$
3: end function
```

The survivability heuristic in Algorithm 5 is a measure of how safe the evacuation plan is. To find the survivability heuristic for one path, the inverse survivabilities for all nodes in the path are summed, and multiplied with the number of evacuees assigned to this path. The survivability heuristic for the whole solution is then simply the sum of the corresponding values for each path.

An alternative algorithm for calculating survivability was also tested. That version calculates the survivability for each path by multiplying the survivabilities of each node included in the path, in essence calculating the survival probability. Though this solution may seem more intuitive, it cannot be used when the value is to be used as a fitness measure, because it cannot provide a fine enough distinction between paths of different quality. For instance, a path with only one lethal node (survivability at 0), and a different path where more than one lethal node is included, would both get a survivability value of 0 using this alternative algorithm. Even though the survival probability for the first path is the same as the second, in the case of the first path it is actually a better path and should have a fitness value which is better than the other path.

3.5.2 Total Distance

Algorithm 6 Total Distance Heuristic

```
1: function TOTALDISTANCEHEURISTIC($C$)
2:     return $\sum_{\mathcal{P} \in C} \sum_{e \in \mathcal{P}} \text{LENGTH}(e)$
3: end function
```

The total distance heuristic in Algorithm 6 is a measure of the total distance the evacuees have to travel according to the evacuation plan. It is calculated by summing the length of all edges which are included in the solution C.

3.5.3 Endpoint Capacity

The endpoint capacity heuristic in Algorithm 7 is a measure of whether there is room for all evacuees in lifeboats, and if not, how many of them there are not room for.

The algorithm for calculating this heuristic starts by initializing the total o to zero. It then goes through all safe nodes in the network. On line 4, G is assigned the set of all groups that are assigned to node n. The next line calculates the total size s of those groups, i.e. the number of individuals assigned to node n. Line 6

Algorithm 7 Endpoint Capacity Heuristic

```
1: function ENDPOINTCAPACITYHEURISTIC($C$)
2:     $o \leftarrow 0$
3:     for every safe node $n$ do
4:         $G \leftarrow$ GROUPSASSIGNEDTONODE($C, n$)
5:         $s \leftarrow \sum_{g \in G}$ GROUPSIZE($g$)
6:         $d \leftarrow s -$ CAPACITY($n$)
7:         if $d > 0$ then
8:             $o \leftarrow o + d$
9:         end if
10:    end for
11:    return $o$
12: end function
```

then calculates the difference d between s and the known capacity of n. If more individuals have n as their destination, d is added to the total o.

3.5.4 Congestion Heuristics

Congestion heuristics in Algorithm 8 and Algorithm 9 is a measure of how much congestion there is.

Similar to the endpoint capacity heuristic, the congestion heuristics start by initializing the total to zero. Then, for every node n, they analyze the nodes or edges for every time step.

At line 7, the node congestion heuristic gets all the groups visiting node n at time step t. Then the calculation proceeds by finding s, the total number of individuals in the groups G. The difference d between s and the capacity of n is then calculated, and if there are more evacuees than there is room for in n d is added to the total o. This process is repeated for every time step t and for every node in the network.

The edge congestion heuristic is similar, analyzing edges instead of nodes.

3.5.5 Grouping

Grouping is a fitness function that measures how much a groups stays together. This is done by counting the number of edges that is overlapped within each group.

Algorithm 8 Node congestion heuristic

```
1: function NodeCongestionHeuristic($C$, $N$)
2:     CalculateGroupTimings($C$)
3:     $T \leftarrow$ the last time step
4:     $o \leftarrow 0$
5:     for $n \in N$ do
6:         for $t$ to $T$ do
7:             $G \leftarrow$ GroupsVisitingNode($C, n, t$)
8:             $s \leftarrow \sum_{g \in G}$ GroupSize($g$)
9:             $d \leftarrow s -$ Capacity($n$)
10:            if $d > 0$ then
11:                $o \leftarrow o + d$
12:            end if
13:        end for
14:    end for
15:    return $o$
16: end function
```

Algorithm 9 Edge Congestion Heuristic

```
1: function EdgeCongestionHeuristic($C$)
2:     $o \leftarrow 0$
3:     for every edge $e$ do
4:         for every time $t$ do
5:             $G \leftarrow$ GroupsTraversingEdge($C, e, t$)
6:             $s \leftarrow \sum_{g \in G}$ GroupSize($g$)
7:             $d \leftarrow s -$ Capacity($e$)
8:             if $d > 0$ then
9:                 $o \leftarrow o + d$
10:            end if
11:        end for
12:    end for
13:    return $o$
14: end function
```

An overlap occurs when atleast two persons use the same edge at the same time step. The returned fitness value will never be 0 (optimum) if people are spread around.

3.6 Super Selector

Unlike traditional GA, NSGA-II does not yield a single solution which can be considered the best one. This is an intended effect of using Pareto ranking. Instead, the solutions present in the first Pareto front (i.e. front 0) are the set of the best solutions which the algorithm could find. Because the solutions with the same rank are mutually non-dominant NSGA-II makes no assumptions as to which, if any, of the objective functions are more important.

Therefore, a single solution must be extracted from the set of solutions yielded by NSGA-II. This can be done manually, which is suitable in a decision-support system. However, automating the process is often preferable, which can be accomplished by adding a final processing step for the set of solutions NSGA-II yields. This can be realized by using a selection mechanism which is able to rank the solutions, for instance by combining the fitness values in some way. Here we use a prioritized fitness ranking approach.

Prioritizing works as follows. Starting with the highest ranking fitness measure, all solutions' value for it is compared. If one solution has a strictly lower value than all others, then that solution is selected. Otherwise, the set of solutions with the lowest value are compared again, this time on the next-highest ranking fitness measure. This continues until a solution has been found. If all objective functions have been processed in this way and more than one solution are still candidates, the tie is broken arbitrarily.

The objective functions we use are ranked in the following order:

1. Endpoint capacity
2. Survivability
3. Passage congestion
4. Room congestion
5. Length

The ordering has been determined through informal reasoning. First, we definitely want every evacuee to be assigned to a lifeboat which has room for him or her. This is the highest ranked objective, seeing as failing to accomplish this is considered a hard failure (certain fatality). Second, we want to minimize the time spent in dangerous spaces. This is measured by survivability, but that value is tied to probability and is therefore not as clear cut. Next comes congestion, which can influence actual survivability and cause evacuees to fail at following their assigned routes. Path length is selected for last, because while it is desirable to have the shortest paths possible, this should never be at the expense of any of the other objectives.

Chapter 4

Discussion

4.1 Evacuation Networks

In this section, we provide three different networks we test using NSGA-II, Dijkstra and Random algorithms and we provide the results for those networks. One network has a single known solution and the others represent primarily represent open space and confined spaces. The networks are tested using the same parameters as much as possible, though individual networks have special features.

4.1.1 Test Network

To validate the output of the genetic algorithm, we use a fixed network with a known solution. As shown in Figure 4.1, Test Network is a grid of nodes where there is a single optimal solution. Nodes at the bottom row contains one group of evacuees each that are supposed to move to the horizontal middle and then upwards until the lifeboat in the end, which in effect forms an upside down T evacuation plan. Survivability on this T is 100%. Rooms outside the T path have less survivability. The middle one or two lifeboats , for odd or even sized grids respectively, should hold exactly the number of evacuees in the grid. Node capacity is set to allow exactly one group in a room at any single time step. Edge capacity is set to allow exactly one group to move from one node to another at any single time step. In effect, there is a single solution to this evacuation scenario which can be used to measure the performance of the genetic algorithm.

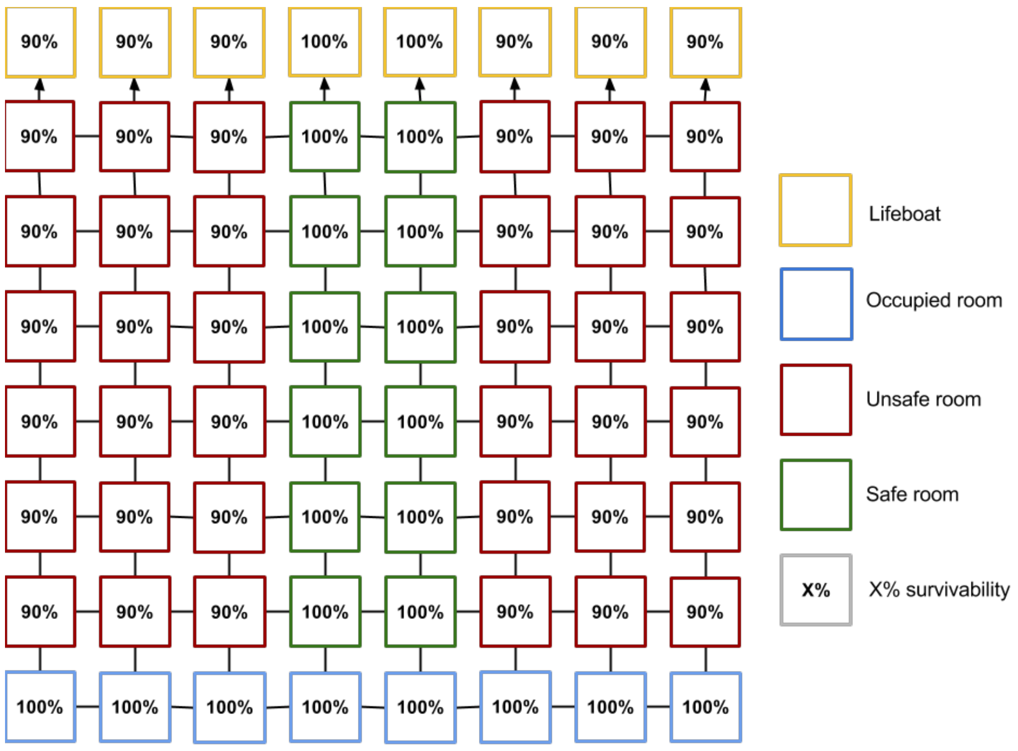


Figure 4.1: A sample Test Network 8 by 8.

Results

Dijkstra, NSGA-II and Random algorithms have been run on this network 50 times each and visualized in Figure 4.2 and 4.3 using a confidence interval of 95%. Parameters have been set to be a strict as possible to yield a single optimal solution. The parameters are as follows used are as follows:

- grid width of 6,
- grid height of 6,
- node capacity of 3,
- edge capacity of 3,
- edge length of 1,
- lifeboat capacity of 5,
- fixed non-path survivability 80% to 100%,
- 18 evacuees divided into groups of size of 3, and
- mutation rate of 0.06%.

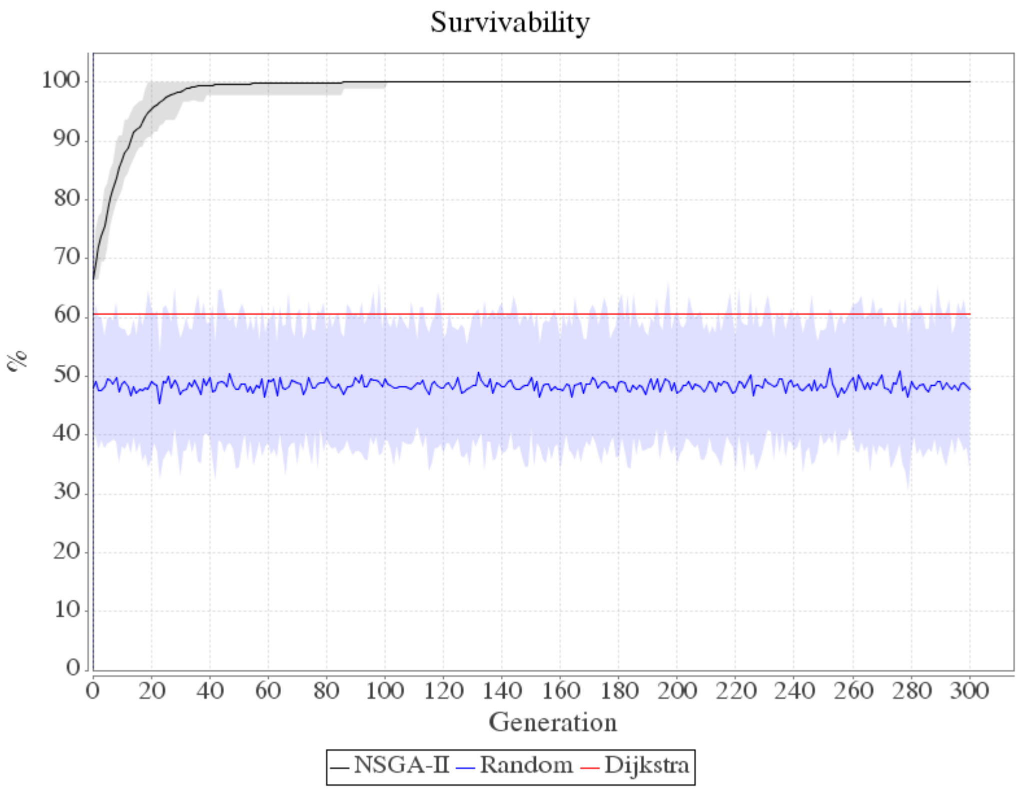


Figure 4.2: Survivability in Grid Test Network with 20 evacuees.

Figure 4.2 shows the survivability of each algorithm. NSGA-II finds the optimal solution after 9 seconds at generation 56. Dijkstra uses the shortest path which is a straight line from each group to the lifeboat resulting in survivability of $\tilde{6}1\%$ in less than 1 second. Random averages 49% survivability.

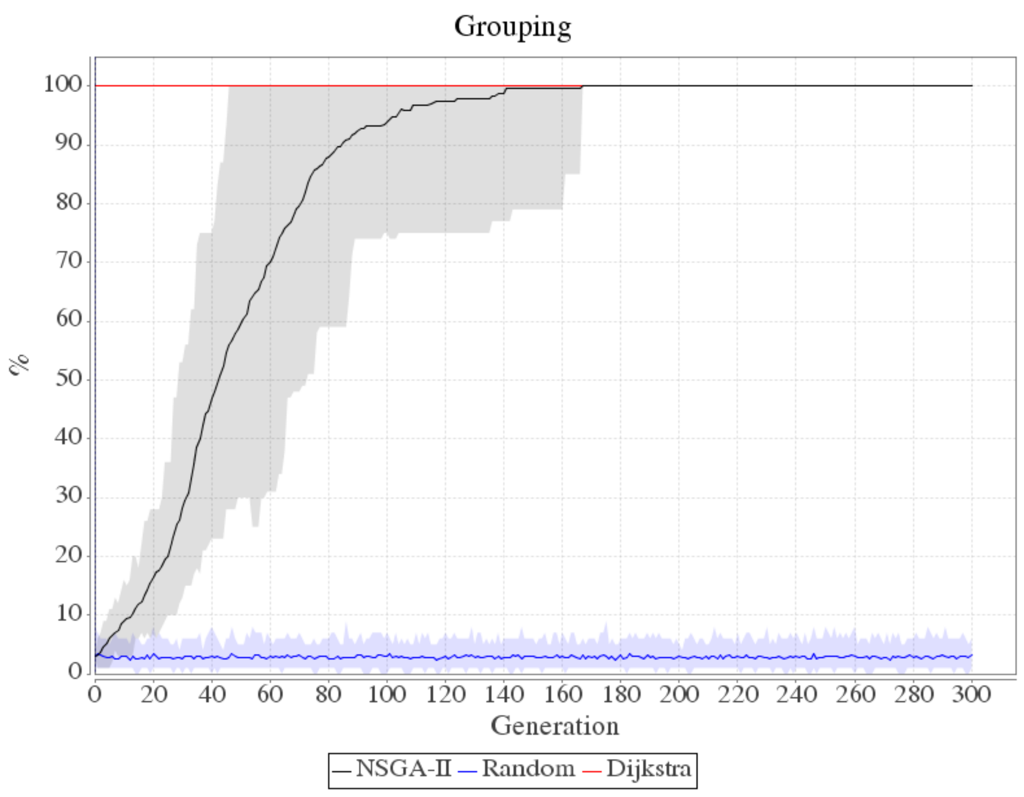
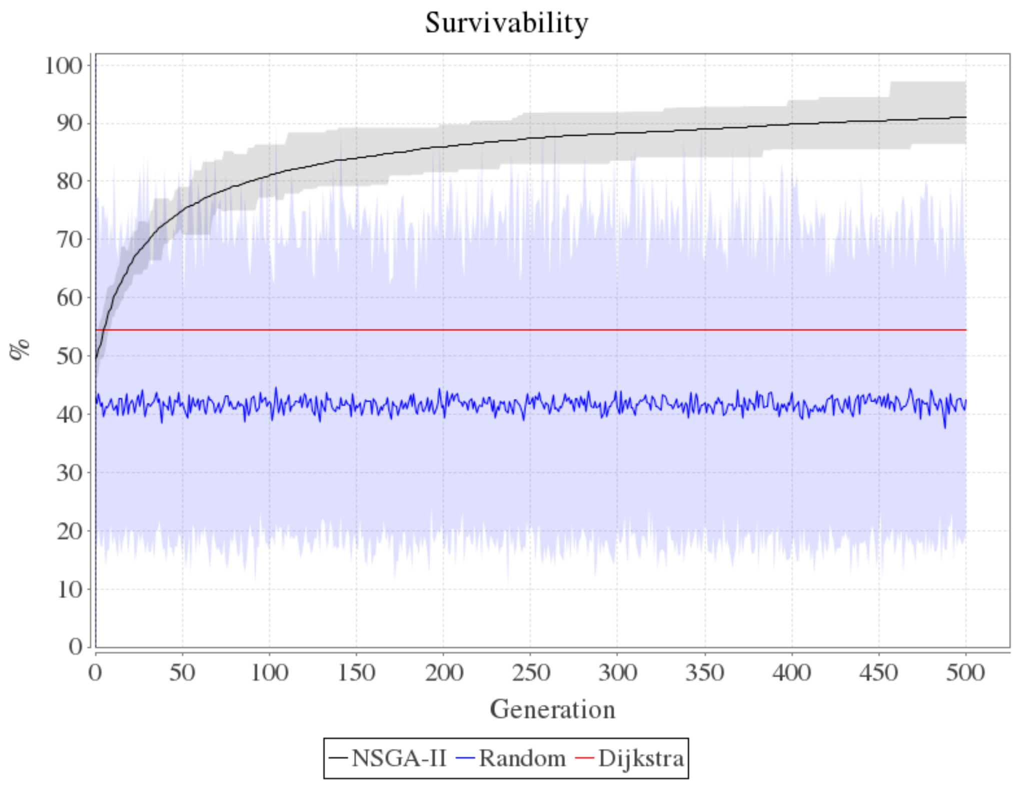


Figure 4.3: Grouping in Grid Test Network with 20 evacuees.

Figure 4.3 shows the grouping of each algorithm. NSGA-II finds the optimal solution after 23 seconds at generation 141. Dijkstra uses the shortest path which is a straight line from each group to the lifeboat resulting in survivability of $\tilde{6}1\%$ in less than 1 second. Random averages 49% survivability.

In the graphs presented in Figure 4.4, the parameters of Grid Test Network has been scaled up to accommodate 100 persons in place of 20, while the size of the network is the same as before. (Note: the graph displayed for the Random algorithm in the grouping chart is an error; performance is closer to the one found in the previous grouping chart). The effect this has is to increase the complexity of the task, as well as increasing the chromosome size required to fully describe the evacuation plan. As can be seen in the graphs, the increased complexity influences the results negatively. Survivability performance is still good, but does not reach the optimal solution within 500 generations. However, the NSGA-II algorithm suffers greatly



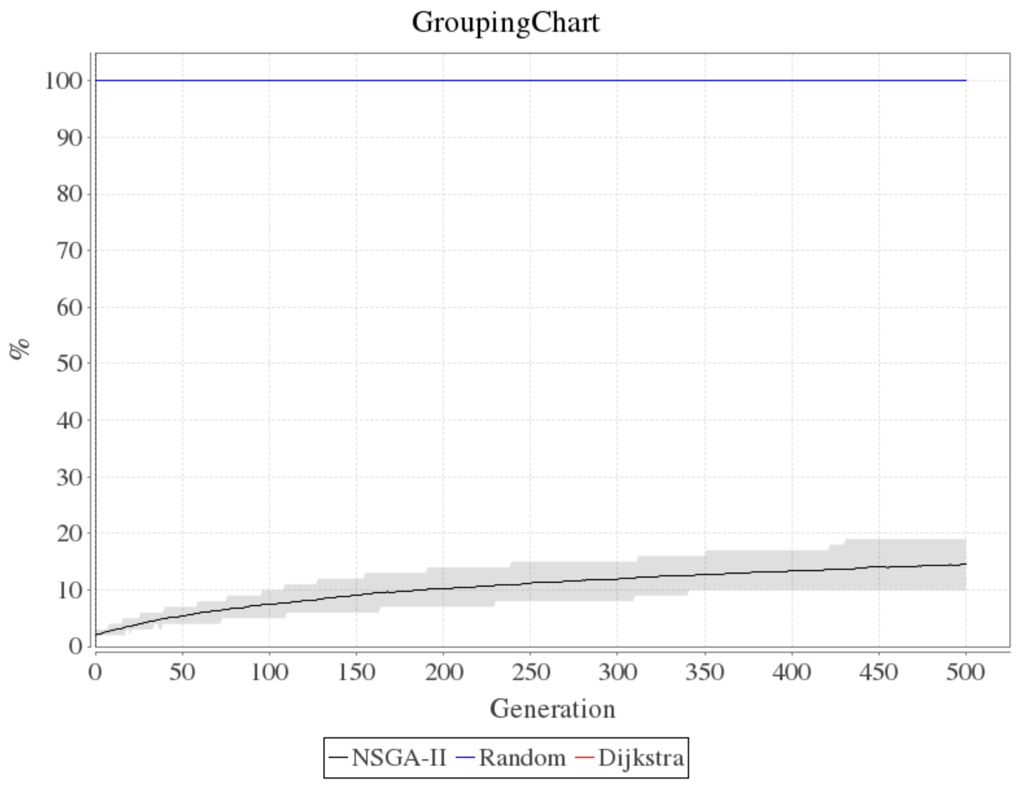


Figure 4.4: Survivability and grouping in Grid Test Network with 100 evacuees.

with respect to grouping. This can be explained by the fact that in this setup, each group consists of many more persons and the algorithm needs to evolve the same path for each group member separately to perform well at this task.

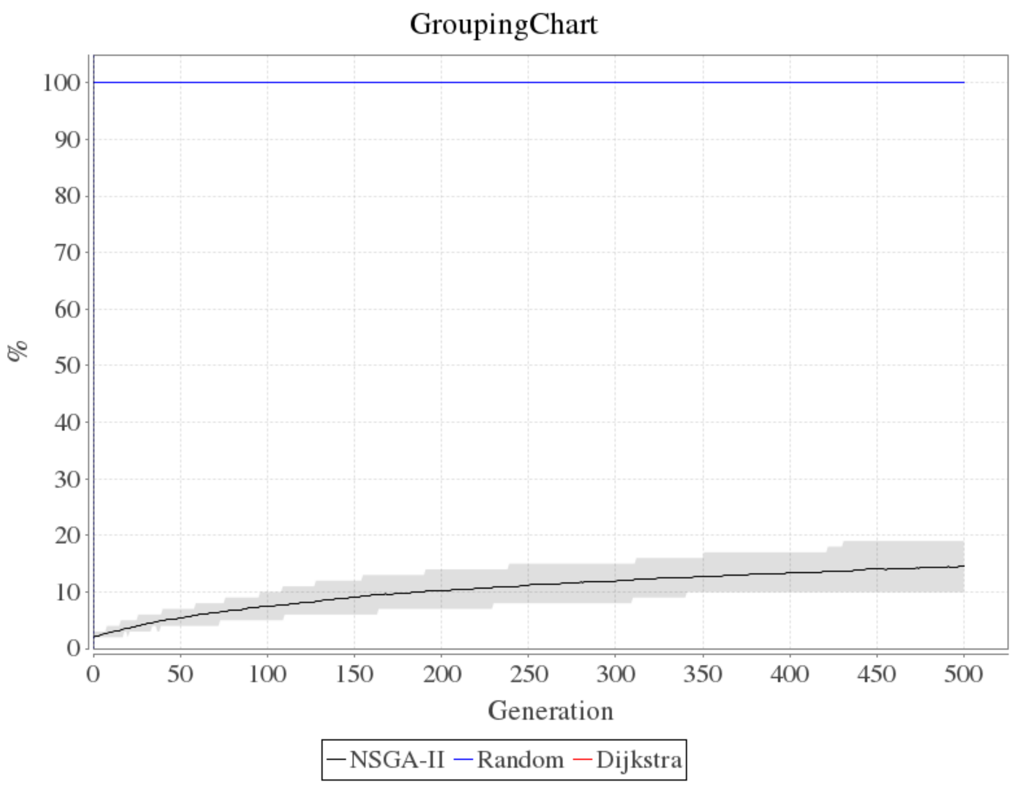


Figure 4.5: Grouping in Grid Test Network with 100 evacuees.

4.1.2 Boat Network

The boat network is based on the layout of a deck on a real ship, the MS Xpedition owned by Celebrity Cruises. The network is illustrated in Figure 4.6.

The symbols in the figure represent the different components of the ship deck, where

- # are empty rooms and hallways,
- `R` are rooms in which people are present,
- `D` are rooms where survivability is less than 1,
- `S` are lifeboats, and
- `|`, `\`, `-` and / are connections between rooms.

```
R R   R R R     #-#-R S   S
| |   | | |    /| | | |   |
#-#-#-#-#-#-#-#-#-D-#-#-#-#-#
    | | | |    \| | |     |
    R R R R     D-D-#-#-#-#-#
                | | | |   |
              D-D-D-# S   S
```

Figure 4.6: The Boat network.

Results

Dijkstra, NSGA-II and Random algorithms have been run on this network 50 times each and visualized in Figure 4.7 and 4.8 using a confidence interval of 95%. Parameters used are as follows:

- grid width of 6,
- grid height of 6,
- node capacity of 10,
- edge capacity of 6,
- edge length of 1,
- random danger survivability 80% to 100%,
- lifeboat capacity of 5,
- 20 evacuees divided into random groups of size of 1-5, and
- mutation rate of 0.06%.

Figure 4.7 shows the survivability of each algorithm. NSGA-II finds the optimal solution of 100% after ˜19 seconds at generation 118. Dijkstra results in ˜94% in less than 1 second. Random averages 90% survivability.

Figure 4.8 shows the grouping of each algorithm. NSGA-II finds the optimal solution after 23 seconds at generation 141. Dijkstra uses the shortest path which is a straight line from each group to the lifeboat resulting in survivability of ˜61% in less than 1 second. Random averages 49% survivability.

4.1.3 Random Network

Random Network is also a grid network like Test Network. The differene is that survivability, location of evacuees are random, location of lifeboats are random, and node capacities and edge capacities have some slack. Survivability varies over time.

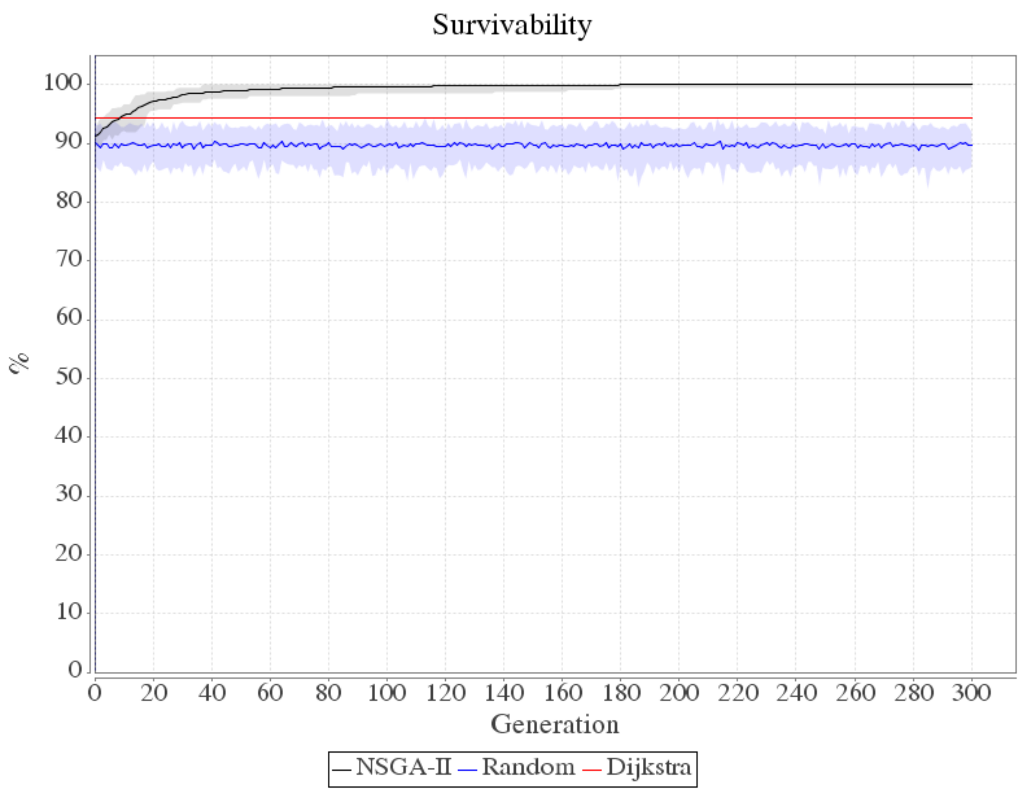


Figure 4.7: Survivability in Boat Network 20 evacuees.

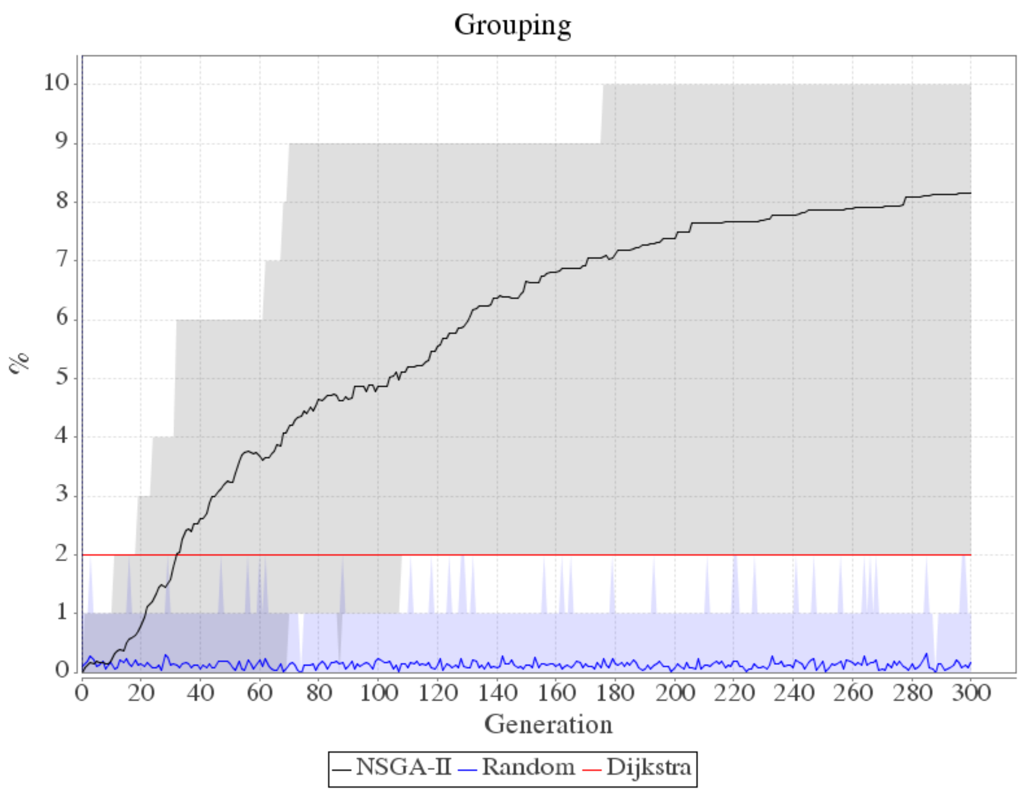


Figure 4.8: Grouping in Boat Network 20 evacuees.

Results

Dijkstra, NSGA-II and Random algorithms have been run on this network 50 times each and visualized in Figure 4.2 and 4.3 using a confidence interval of 95%. Parameters used are as follows:

- grid width of 6,
- grid height of 6,
- node capacity of 10,
- edge capacity of 6,
- edge length of 1,
- random survivability 80% to 100%,
- lifeboat capacity of 5,
- 20 evacuees divided into random groups of size 1-5, and
- mutation rate of 0.06%.

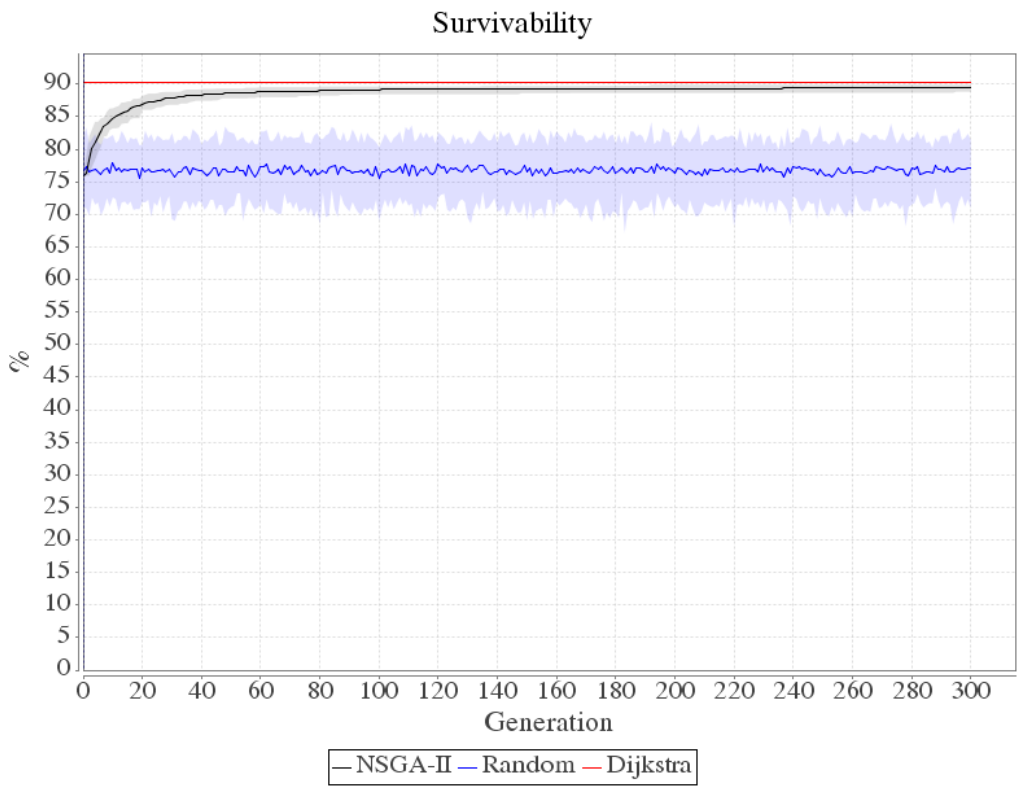


Figure 4.9: Survivability in Random Network 20 evacuees.

Figure 4.9 shows the survivability of each algorithm. NSGA-II seems to converge at 89% survivability using 40 seconds. Dijkstra finds the solution of $\tilde{9}0\%$ in less than 1 second. Random averages at 76% survivability.

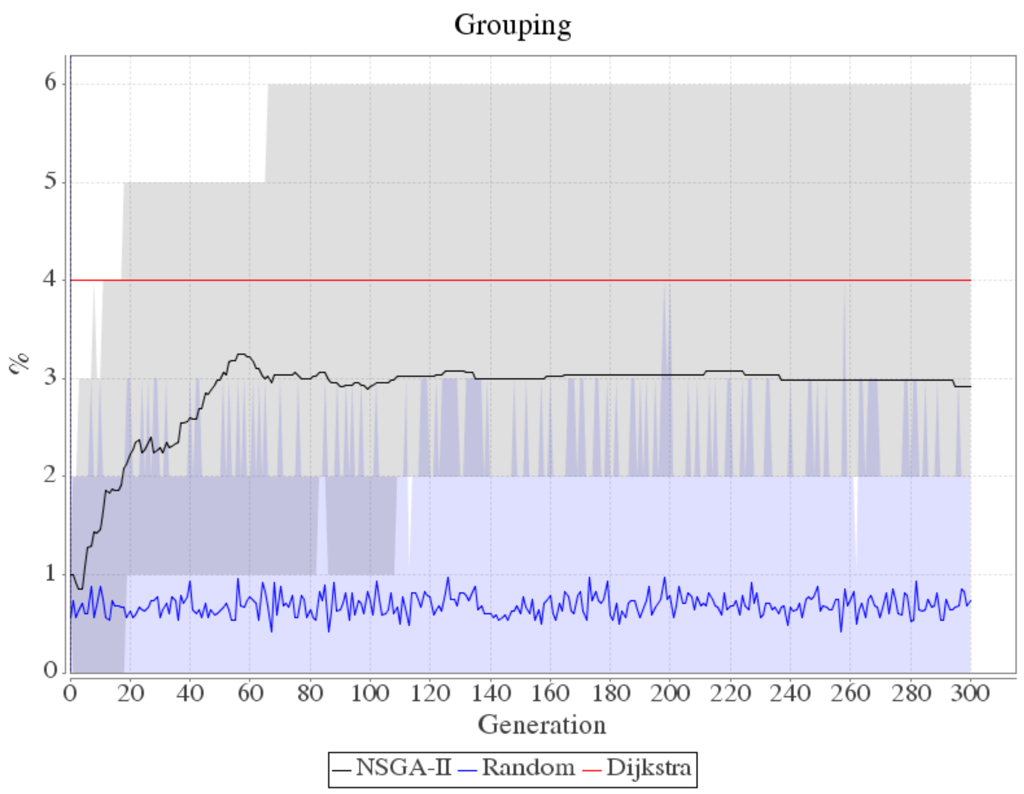


Figure 4.10: Grouping in Random Network 20 evacuees.

Figure 4.10 shows the grouping of each algorithm. NSGA-II performs much better than the random solution. Dijkstra seems to do better, but then again it does not consider congestion.

4.1.4 Effects of Limited Lifeboat Capacity on Heuristics Results

A lack of capacity in the lifeboats presents issues for the heuristics stack. Evacuation success depends on having all evacuees reaching lifeboats. When this fails, human lives can be lost (see for instance the case of the infamous ship RMS Titanic, which was severely lacking in lifeboat capacity). In the prioritized fitness stack, when the endpoint capacity cannot be optimized further, the algorithm goes on to select other solutions where the other goals are improved. However, this does not have the intended effect since, no matter what, evacuees are not able to be rescued due to the lack of free space on lifeboats.

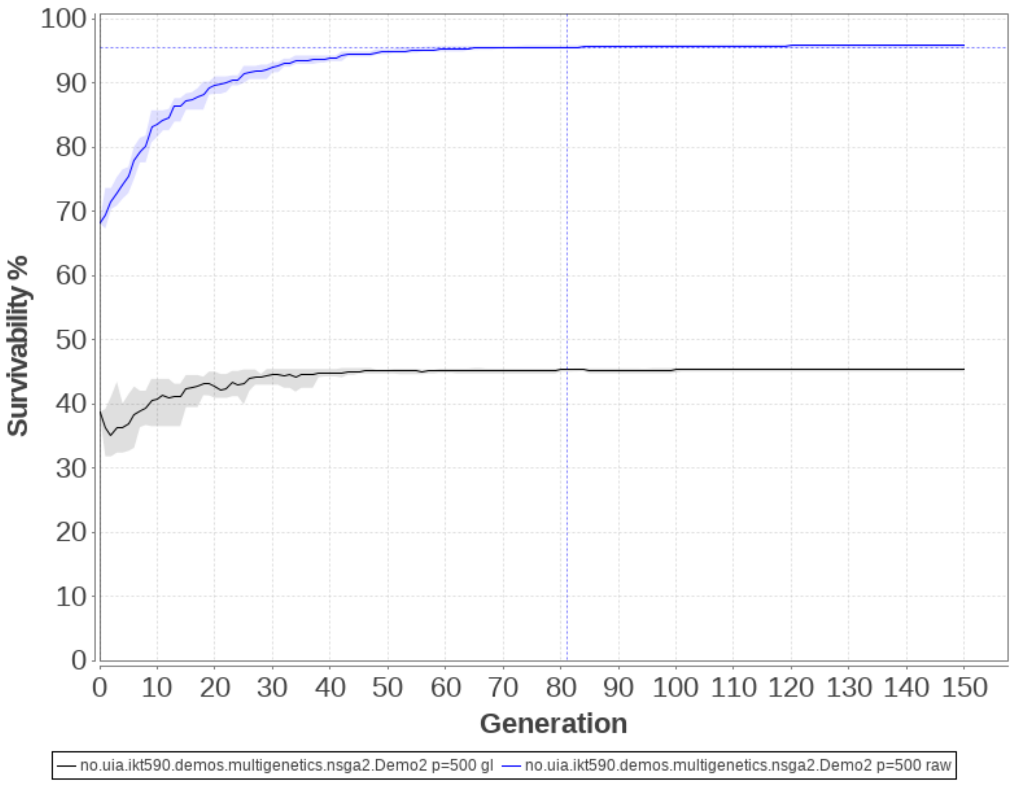


Figure 4.11: A lack of lifeboat capacity.

Consider Figure 4.11, noting the large discrepancy between the graphs. This plot is a result from running NSGA-II on a modification of the 6×6 random network, where there is room for only about half the passengers on the lifeboats. The topmost graph is the survivability as calculated along the best path selected by the algorithm for every generation. This calculation does not take into account anything else than the survivability of each node. Due to the lack of lifeboats, less than half of the passengers are saved.

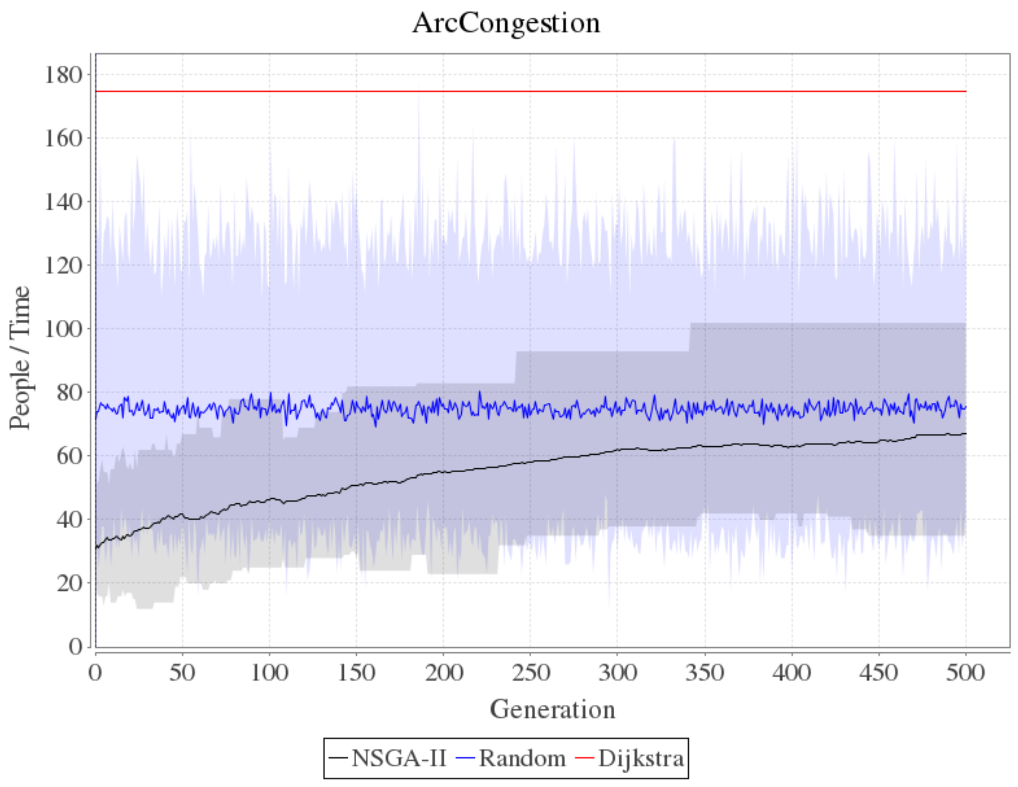
4.1.5 Effects of Congestion heuristic

In Figure 4.12, the algorithms have been applied to the same random 6×6 network with the same parameters. However, for the first graph NSGA-II was run without the congestion heuristics, while it was present for the second. The effect this has can be seen clearly: when not optimizing for congestion, the criterion is clearly neglected and increases as the genetic algorithm progresses. Congestion even approaches the value of the Random algorithm. Conversely, when congestion is optimized for, the genetic algorithm continually improves the congestion performance, albeit slowly.

Dijkstra's algorithm is of little interest here. It optimizes for survivability, and the high congestion value can be explained by Dijkstra routing many evacuees through the same, high-survivability path, without concern for congestion.

The same simulation (including congestion heuristics) is presented with focus on survivability in Figure 4.13. Even though Dijkstra focuses on optimizing survivability, after around 70 generations NSGA-II climbs past, and continues to rise.

Not shown is the impact different fitness function setups have on survivability. In fact, when congestion is not included, NSGA-II manages to beat Dijkstra in less generations, and continues to rise a bit faster than the version where congestion heuristics are included.



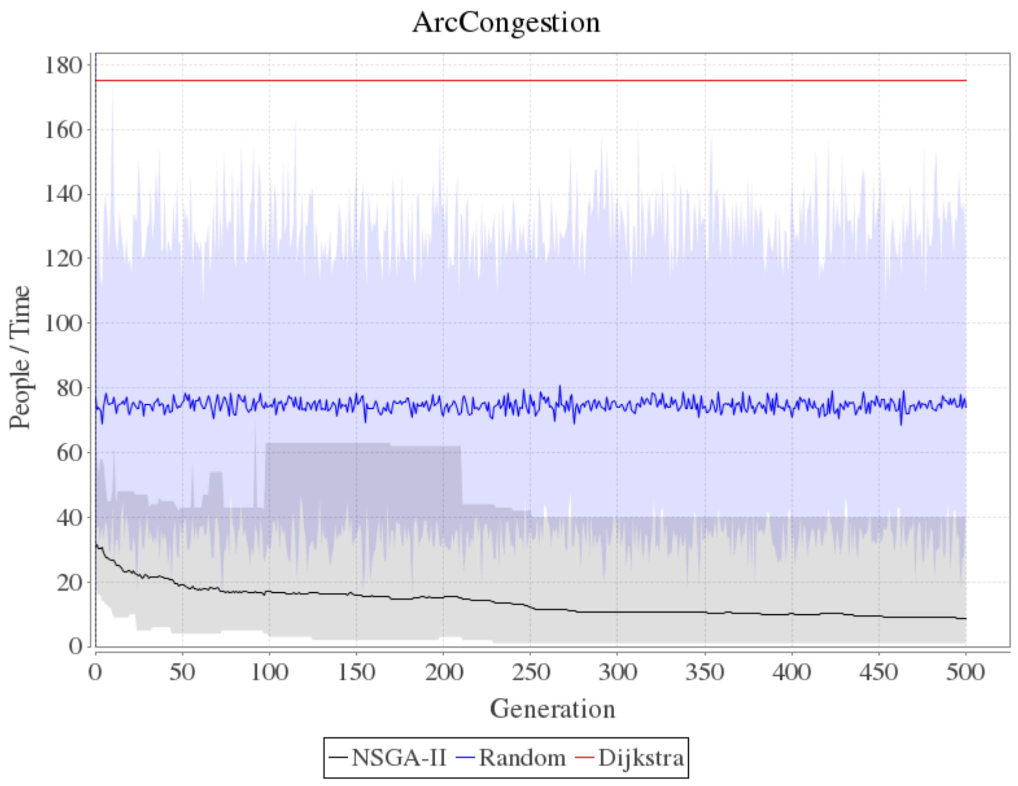


Figure 4.12: Congestion.

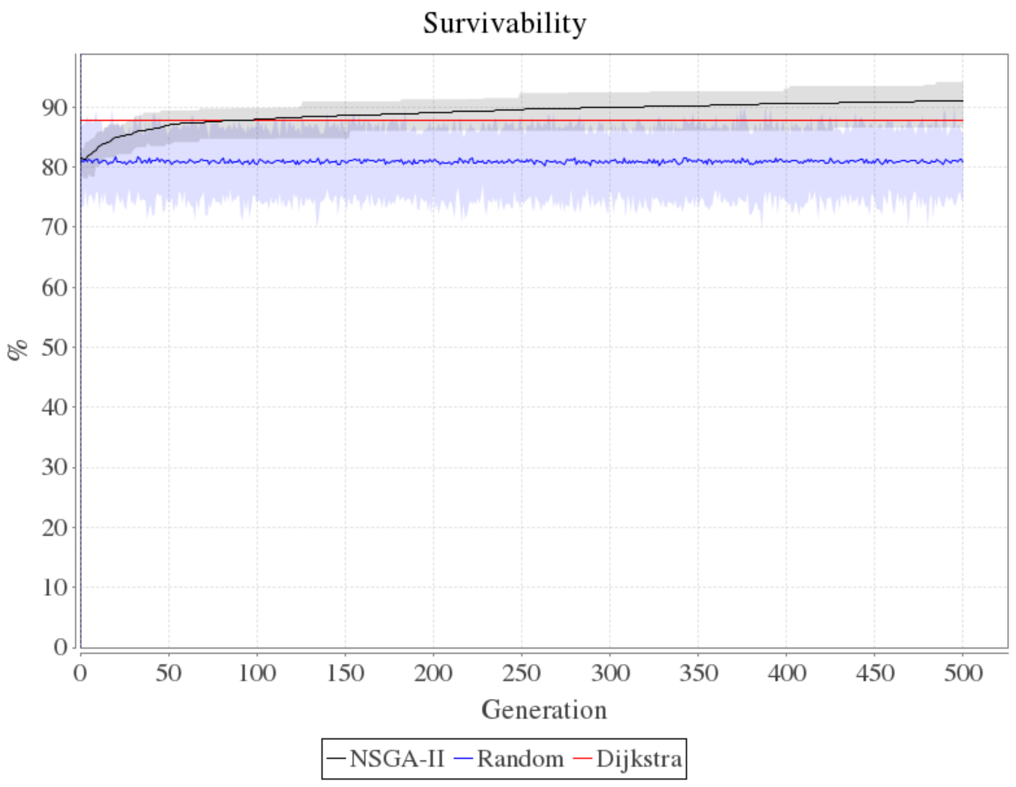


Figure 4.13: Survivability.

Chapter 5

Conclusion

Evacuation during crisis situations is an important task. With efficient evacuation planning and execution, lives may be saved. Equipment such as smartphones and other sensing devices can provide enhanced assistance to both crisis response teams and evacuees, by collecting, processing, and presenting data. The data may be shared via networks, and with access to all this information a system can automatically calculate the best evacuation routes for the current situation.

The problem of efficient evacuation can be viewed as an optimization problem, for which many solution techniques have been developed. Among them, genetic algorithms (GAs). GAs, modeled after natural evolution, is a stochastic method leveraging randomness to evolve incrementally better solutions over many generations.

Regular GAs employ a fitness function to determine the performance of candidate solutions. However, in many problems there are more than one objective to optimize. Naturally, a plethora of methods have been developed to tackle the issue of multi-objective optimization (MOO) problems, and several such methods have been based upon GAs, forming what is called multi-objective genetic algorithms (MOGAs).

NSGA-II, the MOGA we employ, is an adaption of the GA framework which supports the preservation of diversity among candidate solutions by taking into account Pareto indifference. Pareto indifference simply means that, among a set of solutions, no solution dominates (i.e. is strictly better than) the others. This type of MOGA leaves the decision to select one solution among the Pareto optimal ones to an external process which is unspecified and unrelated to the algorithm.

The technique we developed to select a solution from the multiple solutions returned by NSGA-II is a prioritized objective approach. In this technique the objectives are prioritized and a solution is selected based on this.

From analysing the results, we can draw the conclusion that genetic algorithms have the possibility of solving the optimization problem of evacuation planning. We found that in some simpler scenarios the GA variant named NSGA-II was able to find an to optimal solution most of the time, which was a better result the Random algorithm, and partly better than Dijkstra's algorithm.

Drawbacks include that when more fitness functions are included, such as for congestion, the efficiency of the algorithm suffers. We saw specifically that survivability performance went down, both in the number of generations needed and in the achieved values.

5.1 Future Work

Suggested future work include making more specifically adapted genetic operators, for instance mutation operators which take into account the grouping aspect.

The inherent complexity of the chromosome is likely a hurdle which needs to be overcome. Due to the way the chromosome is defined, very specific constraints are applied to it which limits the effectiveness of the genetic operators, compared to traditional genetic algorithms. The limitiations are related to the way each part of the chromosome must be a valid path specification.

Acknowledgements

We would like to thank our supervisors Morten Goodwin and Ole-Christoffer Granmo for their constructive feedback and helpful guidance.

University of Agder, 2013

Bibliography

[1] Maged N Kamel Boulos et al. “Crowdsourcing, citizen sensing and sensor web technologies for public and environmental health surveillance and crisis management: trends, OGC standards and application examples”. In: *International journal of health geographics* 10.1 (2011), p. 67.

[2] Kalyanmoy Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *Evolutionary Computation, IEEE Transactions on* 6.2 (2002), pp. 182–197.

[3] Thomas E. Drabek and David A. McEntire. “Emergent phenomena and the sociology of disaster: lessons, trends and opportunities from the research literature”. In: *Disaster Prevention and Management* 12.2 (2003), pp. 97–112.

[4] Lester R Ford and Delbert R Fulkerson. “Maximal flow through a network”. In: *Canadian Journal of Mathematics* 8.3 (1956), pp. 399–404.

[5] Morten Goodwin et al. “Ant colony optimisation for planning safe escape routes”. In: *26th International Conference on Industiral Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2013*. To be published.

[6] H.W. Hamacher and S.A. Tjandra. *Mathematical Modelling of Evacuation Problems: A State of the Art*. Tech. rep. 24. Fraunhofer (ITWM), 2001.

[7] Abdullah Konak, David W. Coit, and Alice E. Smith. “Multi-objective optimization using genetic algorithms: A tutorial”. In: *Reliability Engineering & System Safety* 91.9 (2006). Special Issue - Genetic Algorithms and Reliability, pp. 992–1007. ISSN: 0951-8320. DOI: 10.1016/j.ress.2005.11.018. URL: http://www.sciencedirect.com/science/article/pii/S0951832005002012.

[8] Sirisak Kongsomsaksakul, Chao Yang, and Anthony Chen. “Shelter location-allocation model for flood evacuation planning”. In: *Journal of the Eastern Asia Society for Transportation Studies* 6 (2005), pp. 4237–4252.

[9] N.D. Lane et al. "A survey of mobile phone sensing". In: *Communications Magazine, IEEE* 48.9 (2010), pp. 140–150. ISSN: 0163-6804. DOI: 10.1109/MCOM.2010.5560598.

[10] Gino J. Lim et al. "A capacitated network flow optimization approach for short notice evacuation planning". In: *European Journal of Operational Research* 223.1 (2012), pp. 234–245. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2012.06.004. URL: http://www.sciencedirect.com/science/article/pii/S0377221712004596.

[11] Anthony R Mawson. "Understanding mass panic and other collective responses to threat and disaster". In: *Psychiatry: Interpersonal and biological processes* 68.2 (2005), pp. 95–113.

[12] J. Radianti et al. "Crowd Models for Emergency Evacuation: A Review Targeting Human-Centered Sensing". In: *System Sciences (HICSS), 2013 46th Hawaii International Conference on*. 2013, pp. 156–165. DOI: 10.1109/HICSS.2013.155.

[13] Mohammad Saadatseresht, Ali Mansourian, and Mohammad Taleai. "Evacuation planning using multiobjective evolutionary optimization approach". In: *European Journal of Operational Research* 198.1 (2009), pp. 305–314. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2008.07.032. URL: http://www.sciencedirect.com/science/article/pii/S037722170800670X.

[14] James MacGregor Smith. "Evacuation networks". In: *Encyclopedia of optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Springer US, 2009, pp. 940–950. ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0. URL: http://link.springer.com/referencework/10.1007/978-0-387-74759-0/page/1.

[15] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. "Comparison of multiobjective evolutionary algorithms: Empirical results". In: *Evolutionary computation* 8.2 (2000), pp. 173–195.