Tampere University

Mikko Saavalainen

GENETIC ALGORITHM FOR BEACON PLACEMENT DESIGN

Information Technology and Communication Sciences
Master's Thesis
December 2020

ABSTRACT

Mikko Saavalainen: Genetic algorithm for beacon placement design
Master's Thesis
Tampere University
Master of Science (Technology)
December 2020

The placement of navigational beacons to provide localisation is an active area of research with a long history. It presents problems for automated design due to the combinatorial expansion of the problem with increased complexity, as well as the requirement to factor in multiple aspects of design, such as localisation accuracy and hardware cost.

The main goal of this thesis is to present and evaluate the usage of genetic algorithms to automatically design beacon placements. Additionally, the well-seen metric, its usage in optimisation, and how well it compares to HDOP is presented.

Results showed the efficacy of genetic algorithms for solving this problem, with improvements over a hill-climbing algorithm, as well as a random generation algorithm. The well-seen metric that was used during optimisation provided a computationally lighter alternative to HDOP, while also providing a minimisation of HDOP during optimisation.

Keywords: Genetic Algorithm, Localization, Range-based, Speciation, Well-seen, Horizontal Dilution of Precision

TIIVISTELMÄ

Mikko Saavalainen: Geneettinen algoritmi lähettimien sijoittelun suunnitteluun
Diplomityö
Tampereen yliopisto
Tietotekniikka
Joulukuu 2020

Lähettimien sijoittaminen sijainnin määrittämistä varten on aktiivinen tutkimusalue jolla on pitkä historia. Näiden lähettimien automaattinen suunnittellu on ongelmallista kombinatorisesta kompleksisuudesta johtuen, joka etenkin tulee esille suunittelu ongelman laajentuessa. Lisäksi muiden asioidien huomiointi suunnittelussa, kuten lokalisaation tarkkuus ja laitteiston kustannukset, laajentavat ongelmaa.

Tämän diplomityön pääasiallinen tavoite on tutkia ja esittää geneettisten algoritmien käyttö lähettimien automaattiseen suunnitteluun. Lisäksi hyvin-nähty metriikka, sen hyödyntäminen optimoinnissa ja kuinka se vertaa HDOP metriikkaan, ovat tarkastelussa.

Tulokset esittivät geneettisten algoritmien sopivuuden ongelman ratkaisuun. Geneettisillä algoritmeilla saadut tulokset vertasivat hyvin muihin algoritmeihin, kuten mäen-nousuun ja satunnaiseen generointiin. Hyvin-nähty metriikka, jota käytettiin optimoinnissa, oli laskelmallisesti kevyempi ja aiheutti HDOP metriikan madaltumisen optimoinnin aikana.

Avainsanat: Geneettinen algoritmi, Lokalisaatio, Etäisyyteen pohjautuva, Lajiutuminen, Hyvin-nähty, Tarkkuuden laimennus

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

LOS	Line of sight
GA	Genetic algorithm
GDOP	Geometric dilution of precision
HDOP	Horizontal dilution of precision
WS	Well-Seen
WSN	Wireless Sensor Network
FOV	Field-of-view

1. INTRODUCTION

Localisation, in which the location of some object is determined, and the techniques used to achieve it are a diverse field of study with a long history. Different use cases, such as outdoor or indoor, and the distinction between internal and external localisation provide a wide berth for finding the right solutions to the specific use-case. The actual techniques used, such as range-based, direction-based, etc. will further limit the design and optimisation of the system.

In this thesis, the focus will be on range-based, line-of-sight navigation systems, where a large number of beacons provide range information about some target. The placement and geometry of these beacons can have a large impact on the accuracy of localisation, and other factors such as minimisation of hardware, ease of maintenance, etc. can make designing the placements of beacons a complex task. This often means that expert knowledge is required for design of an effective localisation system.

Automating this design process with the use of various algorithms is an on-going area of research for many types of localisation systems. We will be taking a closer look at the use of genetic algorithms for designing an optimal beacon placement as well as comparing it to other algorithms. The *well-seen* metric will be used as a computationally lighter alternative to the more rigorous *Horizontal Dilution of Precision* (HDOP) metric, which traces its origins to terrestrial and space-based GPS systems.

1.1 Thesis goals

HDOP is a widely used metric and determines how the geometry of beacons will affect the accuracy of any location measurements with the system. While it is a very useful indicator of the optimality of a beacon geometry it is computationally quite heavy to calculate, which can quickly become problematic when dealing with the large number of solutions that need to be evaluated within heuristic algorithms. While heuristic algorithms avoid searching the entire problem-space to provide an optimal solution, they do evaluate a large number of solutions. This is especially the case with genetic algorithms when the population size is large, and the number of generations required to attain a solution increases in relation to the number of available positions and the number of beacons to be placed.

The well-seen metric will be used as an alternative to HDOP during optimisation. Unlike HDOP it does not fully evaluate the geometry of all visible beacons, but instead relies on simply determining whether a point is inside the convex hull of those beacons visible to it. One of the goals of this thesis is to evaluate how closely the well-seen metric mirrors HDOP during optimisation. That is, during automated beacon placement with various algorithms, does the usage of the well-seen metric as an indicator of solution optimality correspond with an improved HDOP result?

One of the algorithms that the well-seen metric can be used in is a *genetic algorithm*. GA is a meta-heuristic algorithm that can be used in a variety of situations where optimisation is required, and a quantitative value is definable for the goodness of a solution. The main goal of this thesis is to evaluate the usage of a genetic algorithm for the automated placement of beacons. A closer look will be taken at various modifications to the genetic algorithm and the various operators that it used based on previous usage in similar optimisation problems in existing research. The core question with these actions is whether a genetic algorithm can be tailored for this problem-area to provide improved results compared to a generic variant?

1.2 Authors contributions

The usage of problem-specific modifications to a generic implementation of a genetic algorithm were mostly based on existing research. However, within the implementation of speciation, the usage of normal distribution to allocate species sizes based on the average number of beacons used by a selected population was unique to this thesis. Its efficacy and various configuration values for the normal distribution are evaluated.

Additionally, the usage of the well-seen metric in a broader optimisation situation with continuous non-discretized beacon placements as well as an evaluation of its relation the HDOP metric is presented in this thesis.

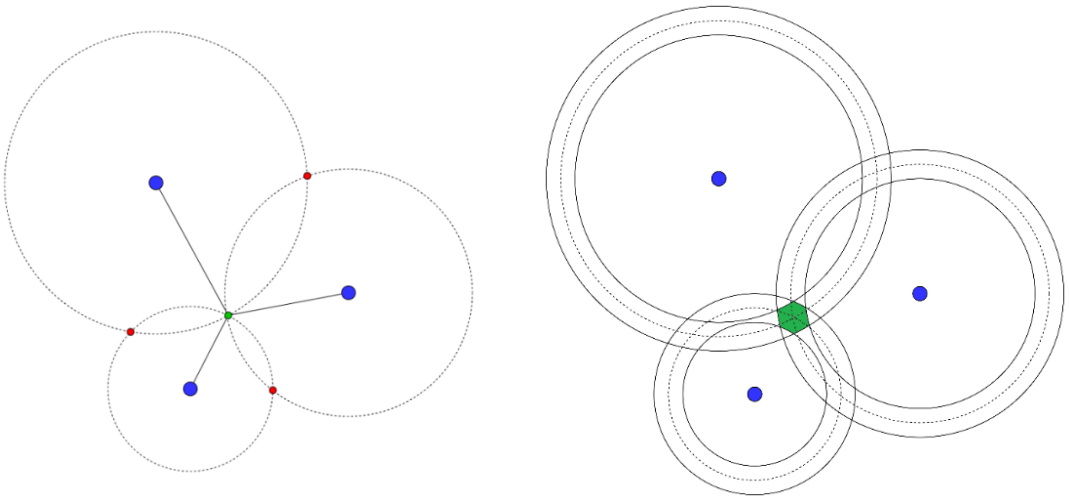
2. PROBLEM DEFINITION

Designing navigation system installations to provide localisation for a specific area requires consideration of a wide variety of factors, such as providing the required localisation accuracy, minimising the amount of hardware that needs to be placed, accounting for reliability in the case of hardware failure, etc. The search space of this problem quickly increases when various additional factors are accounted for such as non-discretized placement of navigational hardware and limited *field-of-view* (FOV), which limits the vision of a beacon, and thus causes the direction any beacon places to be a factor for optimisation. It comes as no surprise that it remains an active area of research for multiple different applications.

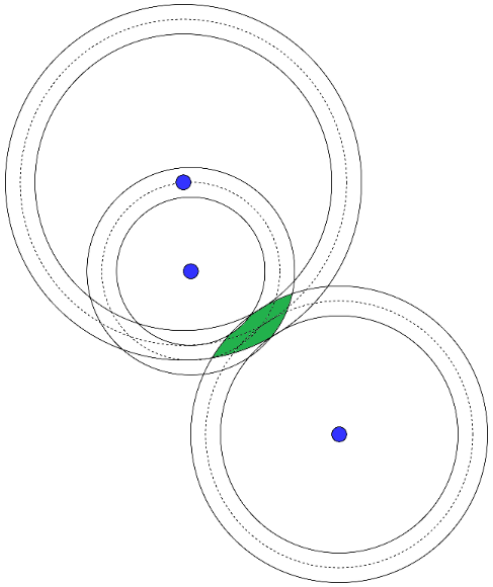
2.1 Localisation with Line-of-Sight beacons

A navigational system based on *line-of-sight* (LOS) by definition requires a direct view between the object being localised, and the devices that distance is determined based on. These devices are referred to by a wide variety of different terms depending on the scientific field, problem-area, and hardware specifics. In this paper the term ***beacon*** will be used to refer to the devices based on which measurement information will be determined, and the term ***receiver*** for the device situated on the object being localised. This is similar to usage in other papers in the same problem-area [1, 2]

From the point of view of this thesis it does not matter whether a beacon or receiver processes the information. The methods presented should be applicable to any LOS system that has beacons placed in known positions and uses range measurements from these beacons to determine the position of the receiver. The position is determined using the technique of *multilateration*, which is an expansion of *trilateration* that uses three points to determine position. With multilateration *n* measurements can be used to determine the position. Figure 1 show various examples of trilateration with different errors and placement positions



a) b)



c)

Figure 1: a) Trilateration, 3 distance measurements b) Trilateration, 3 distance measurements with errors c) Trilateration, 3 distance measurements with errors and poor placement

In Figure 1(a), each beacon marked in blue has taken a distance measurement which is represented by the surrounding circle. Since no directional information is measured, a single measurement only provides a circle on which the point lies. With two measurements the position can already be limited to two points, that is the two points where the circles intersect. These points where two circles intersect are marked in red. Finally, with three measurements a single point can be determined, which is marked in green.

However, in the real world there is no such thing as an errorless measurement, and the impact of internal errors in each measurement on the final result is presented in Figure 1(b). With errors applied, each measurement is now a ring defined by the area enclosed by the two concentric circles around each beacon. Previously, with ideal measurements we could determine a single point, but now we can only determine that the point is somewhere inside the area defined by the intersection of all the rings, marked in green.

Due to the final accuracy being a composition of all the measurement rings, the placement of the beacons will also impact localisation accuracy, and this is presented in Figure 1(c). The position of the beacons now causes a widening and increase of the intersectional area. This means a decrease in the localisation accuracy. When a beacon can no longer provide any measurement at all – either due to the loss of the beacon, or a LOS obstacle blocking the view – this will also change the geometry of the beacons.

2.2 Optimality criterion

Designing a LOS based localisation system is a process that must consider a variety of different factors, ranging from operationally critical requirements to more broad requirements that depend on the specific application.

The key criterion that is being optimised in this paper is the localisation accuracy of the system. As explained in the previous chapter, the geometry of the beacons will impact this, and we will attempt to minimise the intersection of the measurements. One representation of this uncertainty in position is the Horizontal Dilution of Precision (HDOP) [3]. We will later provide a more detailed description of HDOP and its calculation, along with other metrics that provide representations of localisation accuracy.

Reliability can be provided by designing the geometry with redundancy in mind. This means that beacons can be lost – due to physical problems, software problems, a large increase in measurement error, etc. – while remaining in whatever limitations that are imposed on the localisation accuracy. HDOP can be calculated while taking redundancy into account but this is out of the scope of this thesis.

Cost should always be minimised while still fulfilling all other requirements. The total lifetime cost of is be made up of multiple different factors such as hardware, installation, maintenance, etc. However, in this paper we will be considering only hardware and installation.

If the system is organised as a Wireless Sensor Network (WSN) the connectivity between beacons and the power usage of individual beacons and receivers can be important to consider during the design process [4]. In this thesis we will not be taking these into

consideration and will instead be assuming that the beacons have perfect connectivity to each other – through wired connections or some other methods – and power usage is not a limiting factor.

2.3 Optimisation complexity

Methods used to find the globally optimal solution to the beacon placement problem can have very high computational complexity. To illustrate this, a simple beacon placement problem and some example solutions are shown in Figure 2.

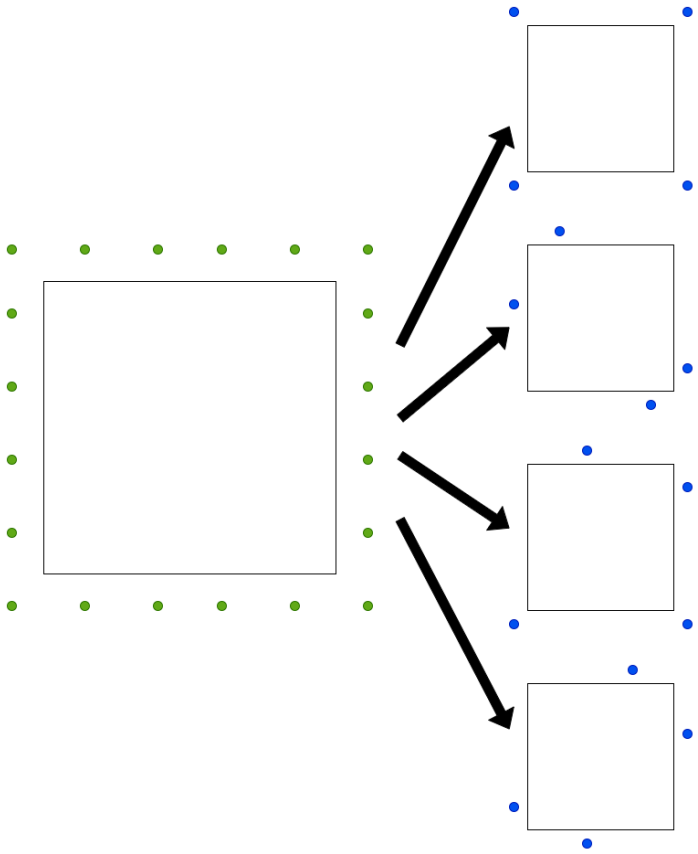


Figure 2: Beacon placement problem and example solutions. Green dots refer to the possible placement positions and the blue dots to placed beacons

In this problem we have a rectangle representing the area where localisation is required and the green dots representing available placement positions. Some example solutions are shown where beacon locations are marked in blue. There is a large number of different ways the beacons can be placed with reference to each other. If the beacons were unique there would be even more ways to place the beacons, but in this case, we can treat the beacons as non-unique. Thus, the problem can be treated as a combinatorial

optimisation problem, where the number of different solutions to any problem of this type is defined by

$$number\ of\ solutions = \binom{j}{k} = \frac{j!}{(j-k)! \times k!}, \quad (1)$$

where j is the possible placement positions, and k is the number of that need to be placed in any combination. The beacon placement problem is an example of a combinatorial explosion and attempting to brute-force it is unrealistic for even limited examples. For example, finding the optimal way to place 4 beacons with the placement positions given in Figure 2 would give us 4845 solutions to evaluate. Now, this does not seem too large of a problem yet, but even if we only double the placement positions to $k = 40$ and attempt to place $j = 8$ beacons, the number of unique solutions would already reach ~76 million. In Figure 3 the combinatorial equation from Equation 1 has been graphed with a fixed number of beacons, but with an increasing number of placement positions.

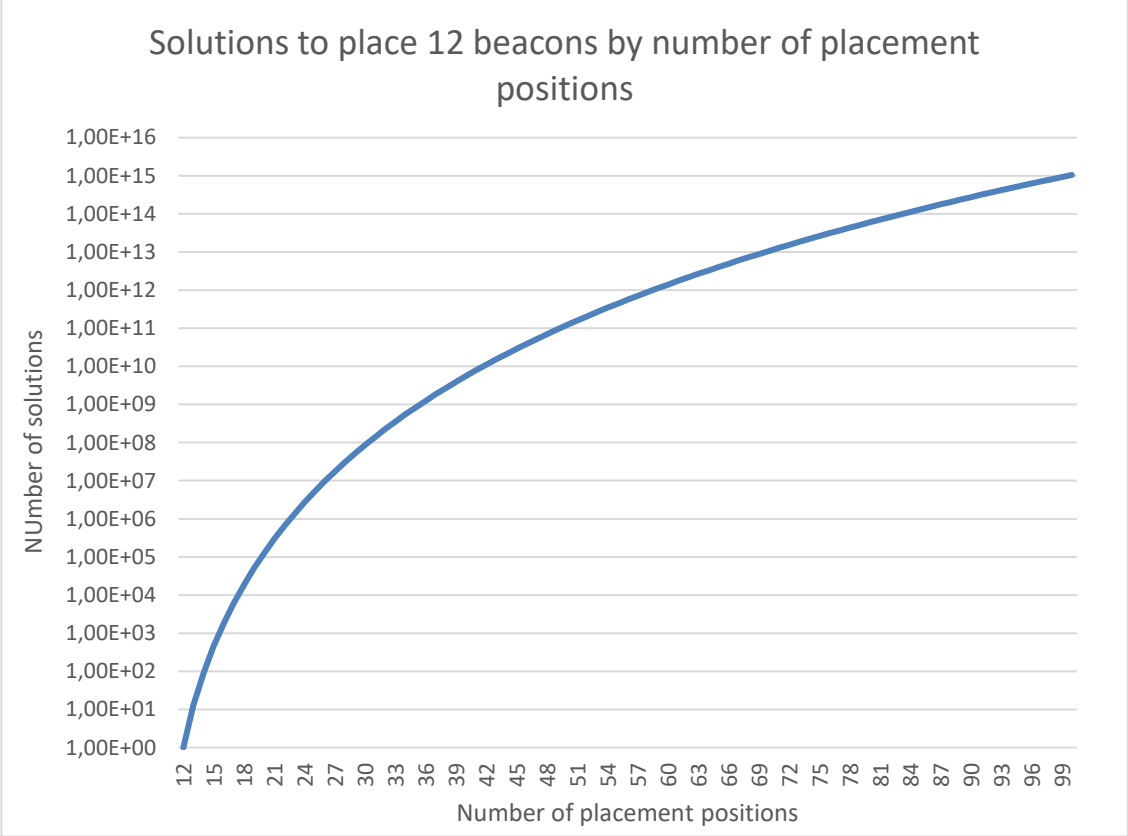


Figure 3: Solutions to place 12 beacons by number of placement positions

As can be seen, the number of solutions quickly increase with the addition of placement positions. Having 100 placement positions is still a relatively simple problem in comparison to actual cases encountered during beacon placement design. Even so, in that case the number of solutions is already more than a quadrillion. Even using an ambitious

evaluation time of 1 $ns/solution$, exhaustively evaluating all solutions would take ~31 years.

While these factors already create an unrealistically large search space for even somewhat large problems, additional factors make it even worse. As beacons may have a limited FOV, their directionality will have an impact on solution optimality, which will cause an expansion of the search space if this were considered. If beacons are required to be placed in an area instead of fixed locations, we can either discretize the area, or place beacons freely. Discretizing will require a good enough resolution, while free placement will make searching the entire search space entirely impossible. Additionally, when a genetic algorithm is used for optimisation the number of beacons is not actually predefined but is instead minimised by factoring it into fitness evaluation which determines the optimality of a solution. This will cause the search-space to increase even further, as evaluating it entirely would require evaluation of how each number of beacons can be placed into the given positions.

Thus, it is clear that brute-force solutions or any solutions that must evaluate even a small portion of the search-space are not viable and it is necessary to turn to heuristic algorithms. This is especially the case since we will be using non-discretized placement for our genetic algorithm in this thesis, in which case the resolution and complexity would approach infinity.

2.4 Existing research

Automating beacon placement design for localisation applications has a long history of research. The solutions proposed over the years involve traditional, gradient based, heuristic, and meta-heuristic algorithms.

In the early 2000s the importance of beacon placement on localisation applications was identified [5, 6]. This early work focused on evaluation of beacon placement optimality in wireless sensor networks and developed simple algorithms to automate beacon placement. The work was very general, without focusing on any specific localisation method. In addition, they looked at improving existing networks with beacon additions and movements to improve localisation, instead of designing placements from scratch.

Research has progressed into various directions and there is currently a wide range of approaches taken to solve the problem. On the more mathematically rigorous end Moreno-Salinas et al. [7] represent solution optimality with the Fisher Information Matrix, which is closely related to the concept of entropy. A solution to finding the gradient of

this is presented, which allows the use of simple gradient based optimisation algorithms to find a continuous solution.

On the other end the usage of various heuristic and meta-heuristic algorithms has provided results in this specific problem-area as well as similar problem-areas such as wireless network design. Z. Fei et al. [8] evaluated a wide range of algorithms for optimising the design of wireless sensor networks based on various criteria. One the of the algorithms was a genetic algorithm. More specifically in the problem area of this thesis, Domingo-Perez et al. [9] presented the usage of an evolutionary algorithm for multi-objective optimisation of beacon design for an indoor localisation system. This work was expanded upon in further research [10], in which a genetic algorithm was used to design a beacon placement while accounting for additional factors such as coverage, number of sensors, and accuracy. Some expansions to a generic genetic algorithm were presented, such as speciation and tournament selection, and these are also made use of in this thesis.

3. PLACEMENT OPTIMALITY

Given a set of proposed solutions to a beacon placement problem we need to quantitatively rate each, while being computationally efficient. Some common metrics used include coverage, which provides information about beacon visibility, and HDOP, which is a scalar “amplifier” of measurement error. Due to the computational complexity of HDOP, some lighter metric that still closely mirrors HDOP would be useful. The well-seen metric, as presented by Allen et al. [1], will be used as a simpler and more computationally efficient metric for localisation performance.

3.1 Coverage

The coverage of a receiver placed on an arbitrary point can be defined as the number of beacons visible from the given point. This can be calculated by determining the geometric relation between the point the receiver is placed on, and all the beacons, while also accounting for LOS obstructions, beacon range, and directionality.

Of course, a continuous area is different from a single point. In the best case we would be able to derive a continuous function that represents the coverage at any arbitrary point. This is a non-trivial task, and the easiest option is to calculate coverage for an area discretely. This is done as a grid coverage, where we first discretize the area into points, and then calculate the coverage individually for each point. This coverage map can then provide a quantitative measure of placement optimality.

The usefulness of coverage as a representation of optimality is limited though. As hinted at in Section 2.1, the geometry of the beacons can have a large impact on the error of the measurement. Coverage is inherently a simplification of the beacon geometry with locational information lost, thus it is not necessarily in line with actual localisation performance. An expanded version of coverage is however useful in the calculation of other metrics. If we additionally store what specific beacons were visible from a point – instead of just representing them with a total number – we have a map that contains complex geometric information for each point that considers LOS obstructions, range, and directionality.

3.2 Horizontal Dilution of Precision

There is an inherent error in the range measurement produced by each beacon. A range measurement with no errors is called the *true range*. Various physical effects cause errors, and with these included we get a *pseudorange* measurement. As previously shown in Figure 1, these errors compound to create an uncertainty in the determined position. When dealing with a more complex situation – such as in a GPS system – a commonly used metric is the Geometric Dilution of Precision (GDOP). It has been in use in radio signal positioning systems since the 1970's [11] and remains in use in modern GPS systems [3]. It consists of vertical (VDOP), horizontal (HDOP) and temporal (TDOP) components. Of these, we are only interested in the horizontal component as we are dealing with a two-dimensional situation.

HDOP acts as a scalar multiplier on the ranging accuracy of each measurement and can be thought of as a scaling factor of the error caused by the measurement equipment. The case is always that $HDOP > 0$, as when $HDOP = 0$, the measurement error would not exist anymore, and we would have a perfect measurement.

Since it has a direct impact on the measurement error, HDOP is a good metric for determining placement optimality. It is, for example, used in GPS receivers, where various algorithms use GDOP to select the best configuration of satellites for localisation [12]. This application is similar to our problem but limited to a small number of beacons (satellites) and with only a single point requiring localisation.

For a given area, HDOP can be individually calculated for discretized points within it to provide us a HDOP map. It is calculated for each point by using geometric information about beacon visibilities from that point. For the calculation of HDOP values we will use the previously mentioned expanded coverage map, which will provide us with the beacons visible from the point along with their locations. A previous implementation was used for calculating HDOP, so this thesis will only go over the basics of it, without delving too far into the mathematics or implementation specifics.

3.2.1 HDOP computation and issues

As previously mentioned, the computational complexity of HDOP makes its usage during optimisation problematic. To understand why the calculation is so complex, let us take a brief look at the process. We won't derive the entire calculation of HDOP here, but will show some key equations presented by Kaplan et al. [13] on how HDOP is calculated for a single point. The general relationship between the covariance of errors and pseudorange error is defined as

$$cov(dx) = \sigma_x = (H^t H)^{-1}\sigma^2_{UERE}\,, \tag{2}$$

where σ^2_{UERE} is the pseudorange error, dx is a vector defining errors in three-dimensions and time, $cov(dx)$ and σ_x are different notations for the covariance of errors, and $(H^t H)^{-1}$ is a representation of how pseudorange error maps into the components of σ_x. Expanding these components, we begin to see where the computational complexity comes from. In Equation 3 below is shown the covariance matrix σ_x,

$$\sigma_x = \begin{bmatrix} \sigma^2_{x_u} & \sigma^2_{x_u y_u} & \sigma^2_{x_u z_u} & \sigma^2_{x_u ct_b} \\ \sigma^2_{x_u y_u} & \sigma^2_{y_u} & \sigma^2_{y_u z_u} & \sigma^2_{y_u ct_b} \\ \sigma^2_{x_u z_u} & \sigma^2_{y_u z_u} & \sigma^2_{z_u} & \sigma^2_{z_u ct_b} \\ \sigma^2_{x_u ct_b} & \sigma^2_{y_u ct_b} & \sigma^2_{z_u ct_b} & \sigma^2_{ct_b} \end{bmatrix}, \tag{3}$$

where the *trace* – referring to the main diagonal values from the top left to the bottom right – of the matrix contains the values required to calculate GDOP along with its subcomponent HDOP. Since HDOP is two-dimensional, the only components we require are $\sigma^2_{x_u}$ and $\sigma^2_{y_u}$, with the other dimensions along the trace not being relevant to our case. HDOP can then be defined in relation to the pseudorange error with the equation given below

$$HDOP \times \sigma_{UERE} = \sqrt{\sigma^2_{x_u} + \sigma^2_{y_u}}\,, \tag{4}$$

which was defined by Kaplan et al. in their comprehensive work “Understanding GPS/GNSS: principles and applications” [13]

A naive solution for calculating HDOP would involve determination of the covariance matrix and then only using certain values within it. Computation can be reduced by determining only the values needed from the covariance matrix σ_x since HDOP is defined within two dimensions. Calculating the values $\sigma^2_{x_u}$ and $\sigma^2_{y_u}$ will require determination of the unit vectors between the point HDOP is being determined for, and all of the measurement locations, which in our case would be beacons. For a single point, we can generalise the computational complexity of HDOP as $O(l)$, where l is the number of beacons visible from that point. This generalisation is based on the definition for H given by Kaplan et al., where it contains all the unit vectors from the point to each beacon. When σ_x is determined using H, as presented in Equation 2, the only values required will be those for the first two dimensions.

Further expanding HDOP to be determined for each point in the coverage map would then give us a complexity of $O(m * l)$, where m is the number of points in the coverage map.

It may seem like the process of determining HDOP is not that heavy of an operation, and it is true that calculating an HDOP value for a single point is not a large operation. However, in the beacon placement problem we have defined this value would need to be calculated for every single point on our coverage map. The usage of any non-gradient based algorithms would require us to calculate this for a huge number of solutions to determine the best, and especially with a genetic algorithm, each generation would require calculating an HDOP map for possibly thousands of solutions.

3.2.2 As a simple representation of optimality

A map of HDOP values can be a useful tool for determining where the given beacon placement provides good results and what areas are problematic. This type of qualitive comparison is useful but makes comparing solutions a difficult and human process. Having a single value or just a few values to represent the HDOP map would be useful when comparing solutions between various algorithms. Due to the computational complexity of HDOP it will not be used directly by any algorithms in optimisation but will instead be used as a final comparison value between algorithms.

The simplest method would be an average or median of all the HDOP values. While simple and easy to determine, these could both be quite inaccurate representations of true optimality, as extreme variances in a small amount of values could cause large changes. In addition, points with an indeterminate HDOP cannot be used in the determination of the average and it thus it will not be truly representative of the actual localizability in the region.

A possible solution for the first issue would be to determine the mean and standard deviation instead. This would allow a more robust comparison between different solutions, as a low mean with large variability is not necessarily a better solution that a higher HDOP with low variability.

A naive solution to the second issue would be to replace indeterminable HDOP values with some arbitrarily decided large value. This would have a large impact on the median and standard deviation. A better solution would be to ignore the values entirely and represent the HDOP determinability with an additional metric called the *validity ratio*. This metric would be a ratio of the determinable points to the total points and would indicate how valid our actual HDOP measurements are.

3.3 Well-seen and Well-Heard

These metrics were presented by Allen et al. [1] as optimality criterion for designing beacon placements for autonomous robot navigation. The terms *well-seen* and *well-heard* originate from the use of acoustic beacons in the original paper. Acoustic beacons also work on the principle of range-measurement, so these metrics are easily applicable to our problem.

A given point can be categorised as well-seen (WS) if it falls within the convex hull created by the beacons visible to it. A convex hull for a grouping of points is defined as the polygon with the smallest number of points that encompasses all other points in the grouping. This is visualised in Figure 4, where the convex hull for a grouping of points is shown.

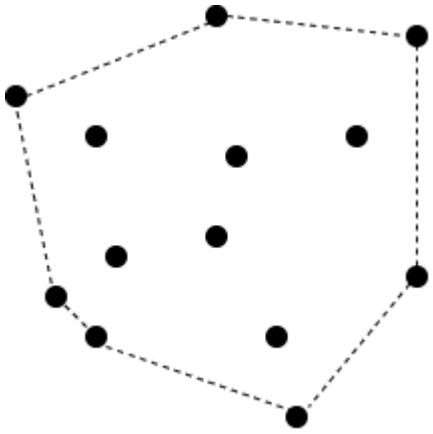


Figure 4: Convex hull for a grouping of points

If a point falls outside the convex hull, but still has enough visible beacons to provide sufficient coverage it is defined as well-heard. This is presented in Figure 5

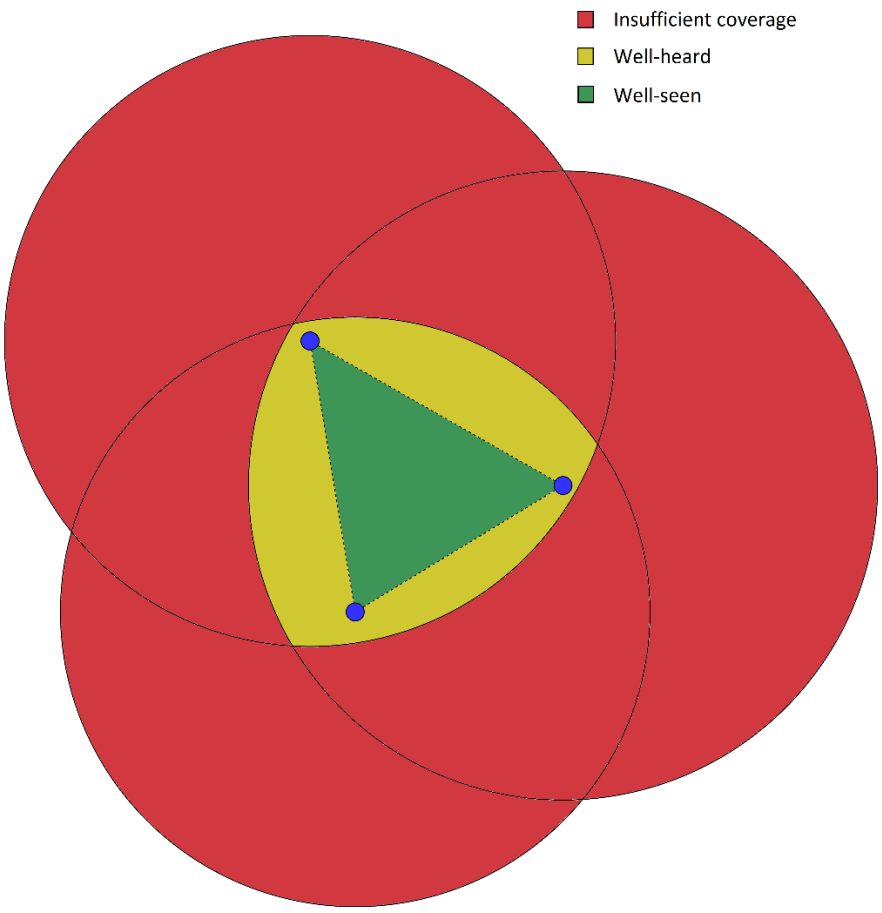


Figure 5: Well-seen and Well-heard visualization. Red representing areas with insufficient coverage, yellow for well-heard areas, and green for well-seen areas

In this situation, each beacon can provide range-measurements within the circle surrounding it. The red areas have less than 3 beacons visible, which is an insufficient amount to provide localisation. Points within the yellow area are *well-heard*, while those in the green area are *well-seen*. *Well-heard* points have sufficient coverage for localisation, but they are not within the convex hull created by the visible beacons. *Well-seen* points have sufficient coverage for localisation and area additionally withing the convex hull created by the visible beacons.

In this thesis only the well-seen metric will be used. While the well-heard metric can provide additional information about areas with un-optimal coverage, we will deal with these in a separate manner that will be detailed later.

A naïve solution for calculation of the WS metric would be to iterate through our coverage map point-by-point, while calculating a convex hull based on the visible beacons and determining whether the point is inside this hull. The convex hull of any set of points can be determined using the quickhull algorithm [14]. This method's computational complexity depends on the specific geometry of the points but can be generalised as $O(l *$

$log(l))$, where *l* is the number of beacons visible from the point. Determining this for each point in our coverage map would then give us a complexity of $O(m * l * log(l))$. Compared to the previously mentioned $O(m * l)$ complexity of HDOP this seems like a poor choice, but with consideration of a few redeeming factors it quickly becomes sensible.

As previously mentioned, HDOP must be calculated for each and every point individually due to its calculation requiring geometric information about how that specific point relates to the beacons around it. However, in the case of well-seen ratio, the convex hull of the visible beacons is the same for each point sharing the same visible beacons, which means it needs to be determined only once for all points sharing the same beacons. This also has the benefit of mostly dissociating the computational complexity from resolution. As the resolution of our coverage map increases there may be new point groupings forming that could not be previously determined due to a lack of points in the edge cases, but in general an increase in resolution will not cause a consistent increase in the amount of convex hulls that need to be determined.

Determining the inclusion of some arbitrary point inside a polygon can be done in multiple ways, but when the points that form the polygon can be assumed to form a convex hull, a complexity as low as $O(log(l))$ can be reached [15], where l is the number of beacons visible from the point. This will then be determined for each point in the coverage map for a complexity of $O(m * log(l))$. Combining this with the previously presented complexity of convex hull determination, we can generalize a worst-case complexity for the calculation of the well-seen metric for a whole coverage map. This complexity is presented in Equation 5

$$O(m * l * \log(l) + m * \log(l)), \tag{5}$$

where l is the number of beacons visible to each point and m the number of points within our coverage map. The implicit assumption in this equation is that the convex hull will need to be determined for each point individually, but as was previously mentioned, the number of convex hulls is generally decoupled from resolution, which gives us Equation 6

$$O(l * \log(l) + m * \log(l)), \tag{6}$$

In addition, if we assume that the number of beacons is significantly lower than the number of points, so $l \ll m$, we can ignore its impact on the total complexity in the portion that represents the calculation of the convex hulls. This then gives us Equation 7

$$O(\mathrm{m} * \log(l)) \tag{7}$$

This compares favourably to the $O(m * l)$ complexity of calculating HDOP for each point in a coverage map and provides our motivation in using the well-seen metric. Intuitively this also makes sense, as with the well-seen metric, in comparison to HDOP, we avoid determining a large number of vectors individually for each point in the coverage map. Both methods do however require information provided by the coverage map, so we will not go into its complexity and implementation as it is a common factor.

Much like in the case of HDOP, a well-seen map will not allow easy comparison between solutions. Similarly, we can attempt to represent the whole with a single value, which due to the binary choice of well-seen – meaning something either is or is not well-seen – is very simple. The ratio of well-seen points to all points gives us a normalised value between 0 and 1, where 1 is a perfect solution and 0 is the worst possible solution.

4. GENETIC ALGORITHM

The combinatorially expanding search-space of the beacon placement problem makes exhaustive search algorithms problematic. A genetic algorithm is an example of a meta-heuristic algorithm, which can provide approximate results more efficiently than exhaustive methods. Problem-specific modifications can lead to further improvements.

4.1 Heuristic and meta-heuristic algorithms

In comparison to exhaustive optimization methods, algorithms belonging to the heuristic or meta-heuristic classes avoid many of the common pitfalls, while also having their own drawbacks. *Heuristic* methods provide quicker results than exhaustive methods by avoiding a full traversal of the search-space, thus providing an approximate solution [16]. They are useful in situations where the exact global optimum is not required, and the desired level of optimality can be reached. The main drawback being that the final solution may not be the exact global optimum, as the problem is not evaluated exhaustively.

A *meta-heuristic* is an expansion of the heuristic concept. While a heuristic is tailored to the specific problem, a meta-heuristic is general and can be applied to multiple types of problems with minimal adaptation. There is a large amount of different meta-heuristic algorithms – but many of the common ones are inspired by natural systems – such as biological evolution in the case of evolutionary algorithms [17], or the physics of annealing in the case of simulated annealing [18].

The beacon placement problem is a good candidate for heuristic and meta-heuristic methods due to a few key characteristics. It is combinatorial in nature, which leads to an exorbitantly large search-space. An exact solution is generally not required, and due to the generic nature of meta-heuristic algorithms, a large amount of knowledge specific to the problem area is not required.

4.2 Genetic algorithm

A genetic algorithm is a meta-heuristic optimization algorithm belonging to the family of evolutionary algorithms, which take their inspiration from mechanisms found in the biological process of evolution. In comparison to single-solution algorithms – which involve one solution being iteratively modified – evolutionary algorithms are population-based [19], meaning there are multiple solutions being simultaneously optimised. This may

avoid convergence on local minima and allows separate areas of the search-space to be simultaneously searched.

At its most basic and without any problem-specific functionality, a genetic algorithm consists of four operations that are iterated through after initialisation, involving *fitness evaluation, selection, reproduction, and mutation.* Before these operations, the algorithm needs to be initialised with a *population* of *individuals*. An individual refers to a data object that contains a solution to whatever optimisation problem is at hand. This solution is referred to as the *genome* of the individual and is comprised of various *traits*. In our case, the genome of a single individual would be a placement of beacons that provides localisation to some predefined area. Each beacon with its position and orientation would be a trait. The genomes of each individual in this initial population are usually completely randomized but can be provided by some other process to give a head start to the iterative process of the genetic algorithm.

After initialisation of a starting population, the operations are iterated through to slowly improve the total fitness. These include *fitness evaluation*, *selection*, *reproduction,* and *mutation*. In Figure 6 is a flowchart showing the general structure of a genetic algorithm

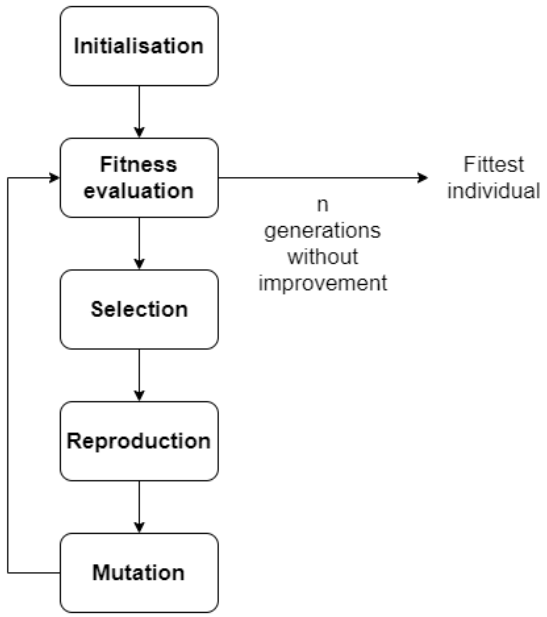


Figure 6: General structure of a genetic algorithm

These operations can be iterated through as long as required, but a general method is to stop after a certain amount of generations are reached without improvement.

4.2.1 Fitness evaluation and selection

Fitness must be evaluated so that individuals and the solutions they offer can be qualitatively compared, after which individuals can be appropriately selected from the total population.

Fitness refers to the performance of a specific individual according to various metrics. While depending on the problem-area it can take on all sorts of values, it does need to be a single qualitative value that can be easily used to compare individuals. At its simplest, the fitness of an individual represents only the raw utility provided by the solution it defines, which in our case would be the well-seen metric. Additional factors such as cost, complexity, or any other problem-specific requirements can be implemented into the fitness calculation to provide a better representation of a individuals' optimality. Equation 5 shows the most basic fitness equation that will be used by the genetic algorithm

$$fitness = WS - c \cdot b, \tag{5}$$

where WS is the well-seen ratio calculated from the geometry of the beacons, c is the beacon cost, and b is the number of beacons used in the solution. With this simple equation we can already optimise with regards to our key metric, the well-seen ratio, and prefer minimising the number of beacons by applying a small cost for each.

When the fitness of each individual in the population has been determined, some portion of the individuals is selected. This is meant to approximate the concept of "survival of the fittest" in evolutionary theory, and only the selected individuals are allowed to proceed to the next phases. In its simplest form, this selection is done entirely based on fitness. For example, one might select all individuals above the 90th percentile. Simple methods like this can have serious drawbacks and can cause significant problems for the algorithm as a whole, such as poor convergence towards optimality due to prevention of long-term individual development, or low selective pressure [20]. Later we present a more problem-specific implementation of the genetic algorithm where we will discuss other methods that can be used to provide a more rigorous selection.

4.2.2 Reproduction

Once a subset of the population has been selected, repopulation is required. Generally, the population size is kept constant between each iteration, so we need to create enough new individuals to bring it back to its previous value. This is done by applying *crossover* between individuals from the selected subset. This process consists of combining the genomes of two individuals to create a new individual with a unique solution compared

to both parents. The genome might not always be unique with respect to the entire population, as even with different parents since the process is stochastic in nature two individuals with identical genomes can be created from different parents. The exact process can vary depending on the implementation, but the end product is always a new individual.

4.2.3 Mutation

The newly generated population is intended to be an improvement on the original population, since only traits from the selected individuals were allowed to pass on. However, none of the previous operations create new traits, but instead just randomly reshuffle traits that were created during initialisation of the starting population. Therefore, mutation is required. Much as with crossover the specific implementation can vary, but the general requirement is that each individual's genome is randomly changed to provide a new solution. This can take the form of modifying, removing, or adding traits.

Applying mutation to an individual can be implemented in various ways and does require some specificity to – and understanding of – the problem-area as we must understand what defines individuality, how to change a solution, and how these impact solution optimality. In our problem we are attempting to place an indeterminate number of beacons onto a two-dimensional area. In addition, the beacons have a limited field-of-view, thus directionality is an extra dimension of that optimization. Taking all these factors into account we can define four mutation operators that can be applied to an individual.

- Translate

  A randomly selected, previously existing beacon has its X and Y coordinates randomly changed. This random change takes the form of adding a randomly generated decimal value from some range $[-a, a]$, where ***a*** is referred to as the *translation range*. A random value is generated and added to both X and Y coordinates separately. If the translated beacon is in an invalid placement, such as inside an obstacle or outside the placement area, translation is retried.

- Create

  A new, validly placed, randomly located, and randomly oriented beacon is added.

- Delete

  An existing beacon is removed.

- Pivot

The orientation of the beacon is changed by adding a randomly generated negative or positive value to it.

How these operators are applied to individuals and the population as a whole can be defined and restricted by various *hyperparameters*. These are parameters that configure the optimisation process. For our case, we only have two of these, the *mutation rate,* and the previously mentioned *translation range*.

Mutation rate defines the probability that a given individual will mutate. With our implementation, only one of the four mutation operators is then randomly selected and applied.

4.3 Problem-specific modifications

The basic building blocks of any genetic algorithm are the four operators described in the previous chapter. To improve both computational performance, and the optimality of the final solution, these basic operators can be implemented and expanded in various ways. New operators can also be made, often borrowing from some additional concepts and structures found in evolutionary theory.

Modifying the existing operators and adding new ones can come at the cost of making the algorithm more problem specific. Some operators and modifications will not provide substantial improvements in all problem-areas, and care needs to be taken when deciding what changes to make to a generic genetic algorithm. There are a few improvements we will be looking at that are of special note for our problem area.

4.3.1 Speciation

In a generic genetic algorithm during each generation there is a single large population containing all individuals. When this population reproduces to create the next generation, any individual can reproduce with any other individual. This can cause problems when solutions that are very dissimilar in their approach are merged to form new solutions that retain none of the traits that were resulting in optimality of each parent.

The solution to this is the evolutionary concept of *speciation* which was first presented in the context of a parallel genetic algorithm by James Cohoon et al. in 1987 [21]. While their approach was done in the context of wanting to easily parallelize the computation of a genetic algorithm, they found improvements to solution optimality in addition to the performance improvements provided by parallelization. Each *species* is used to represents some subset of the entire search space, where solutions are more like each other than to those in some other species. In other words, given a search space, a species represents some limited region in this space.

The actual improvements arise from the limitations we place between species. All operators are applied internally within each species, thus limiting *cross-competition,* which can cause optimization over a wide search space and decrease the ability to converge on locally optimal solutions. Cross-competition refers to individuals with very different solutions having to compete with each other during selection. With multiple species that each are optimizing within a limited search space we can thus achieve local convergence while also avoiding getting stuck in local minima due to different species occupying different *niches*.

This concept can be applied to multiple problem areas, where the major requirement is that the solutions can somehow be grouped based on their closeness with each other. Regardless of the method used to group individuals, the representation of the grouping in the search-space is arbitrary. This is because as we define groupings, we are also defining the search-space on its basis, which will inherently make it a perfect segmentation of the search-space. However, whether any method used in grouping is actually a good representation of the closeness of solutions to each other is not easily evaluated.

Domingo-Perez et al. used speciation [10] in a similar problem setting and managed to achieve significantly improved results in comparison to an unmodified NSGA-II algorithm. Their solution for defining species was based on the number of beacons used in an individual's solution. This is simple, computationally efficient, and has sound reasoning behind it as solutions using the same number of beacons are generally similar in fitness. In addition, when using a single population, bias towards either over or under placement of beacons can easily occur if the cost of placement is not correctly fine-tuned. With speciation, since selection is applied within a species, individuals are not required to compete with solutions that have an unfair advantage towards the bias exhibited by the fitness evaluation.

The usage of speciation in this paper is similar to that presented by Francisco et al. but varies in technical implementation and includes a few additions. Figure 7 shows the general structure of speciation as implemented in this thesis

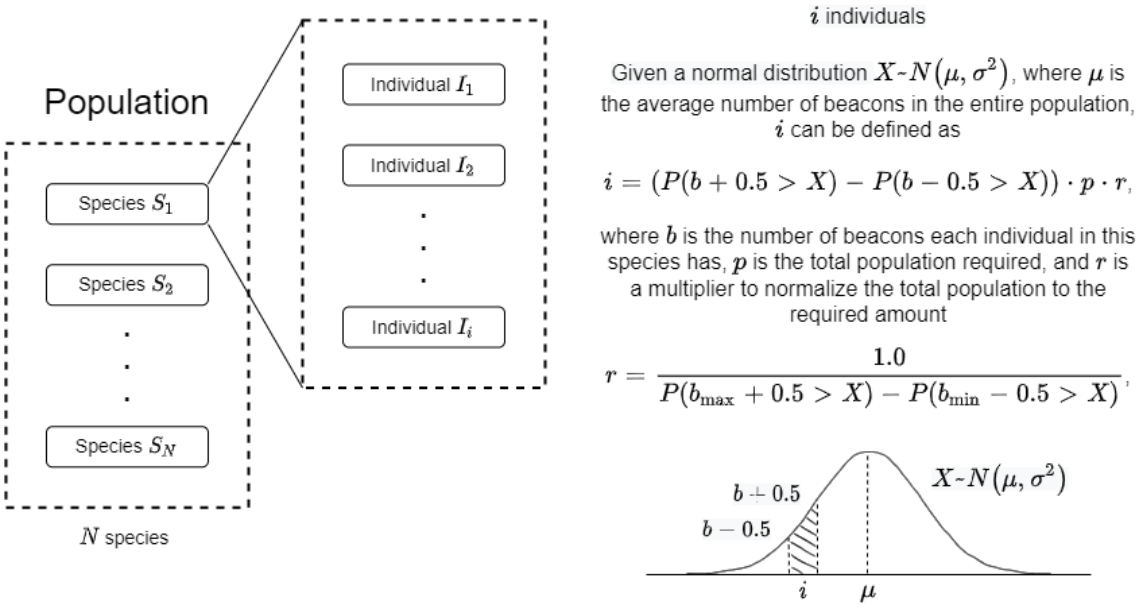


Figure 7: Structure of speciation with species size determined by normal distribution

Given a population of individuals, it is split into N species which each consists of i individuals. This species size i of each species in the whole population is determined with Equation 8 given below

$$i = \big(P(b + 0.5 > X) - P(b - 0.5 > X)\big) * p * r, \qquad when\ X \sim N(\mu, \sigma^2), \tag{8}$$

where b is number of beacons used by each individual in the species, p is the total population required, r is a multiplier to normalize the total population to the required amount shown below in Equation 9

$$r = \frac{1.0}{P(b_{max} + 0.5 > X) - \mathrm{P}(b_{min} - 0.5 > X)}\ , \tag{9}$$

and X follows a standard distribution centred around the average number of beacons found in the *selected* population. An example of this is presented with the parameters given in Table 1 below.

Table 1: Parameters for calculation of species distribution

Population size	Number of species	Selection ratio (s_r)	Beacon average (μ)
1000	7	0.1	7.7

After the selection round is applied, we are left with a population of size *100*, which needs to be bred into a new population of the appropriate size. Each species in this new popu-

lation will be limited to *i* individuals based on the usage of normal distribution as previously described. A species with *6* beacons for example would be limited as shown below in Equation 10

$$i = \big(P(6.5 > X) - P(5.5 > X)\big) \cdot 1000 \approx 240\ individuals, \qquad when\ X \sim N(7.7,1), \quad (10)$$

The benefit of using a normal distribution to determine the size of each species is twofold. Firstly, it allows a larger focus on the species which is currently providing the best results. As the average number of beacons begins to converge on the "optimal" solution, there will be less computational resources wasted on solutions with less or more beacons, that have already not shown to be able to reach the same optimality. Secondly, it allows a wider optimisation while using fewer computational resources, since solutions farther from the average are provided less and less resources.

One additional adjustment made to speciation was the limitation of the shift in the average. Early on after implementation it was noticed that the optimality could easily degrade when the shift to a higher or lower average was too fast. Large sudden shifts will cause species that are not represented in the new average to be discarded entirely and their solutions will not be allowed to develop. The shift was instead limited to *0.2* per generation, meaning that even with a large increase in the average, the distribution of the next generation will be calculated using an average shifted only this amount above the current average beacon number.

4.3.2 Tournament selection

With selection, the goal is to improve the fitness of the population each generation by selecting for strong traits. This is done by selection of some portion of individuals from the total population with a preference towards high fitness. The traits these individuals contain will then be passed on to the next generation during reproduction. How these individuals are exactly selected can have a large impact on the overall performance of the algorithm depending on what kind of selection pressure is applied to the population.

The simplest method for selection involves selecting only the top individuals. The issue with this method is the high selection pressure it causes against individuals with lower fitness. An individual might not have the required fitness to be selected while containing some desirable traits. Divergence from the local minimum should be allowed in the short-term, as otherwise we will converge without allowing solutions in a wider area of the search-space to be evaluated. This is very much a balancing act, and there is a wide array of different methods used to provide selection pressure, while still allowing divergence from the local optimum.

A commonly used method for this is *tournament selection*. In a similar problem setting Leune et al. showed the effectiveness of this method in comparison to a simpler uniform roulette selection method [4]. Tournament selection involves separating the population into groups in which the individual with the highest fitness will have the highest chance of winning. A diagram representing the structure of tournament selection is shown in Figure 8

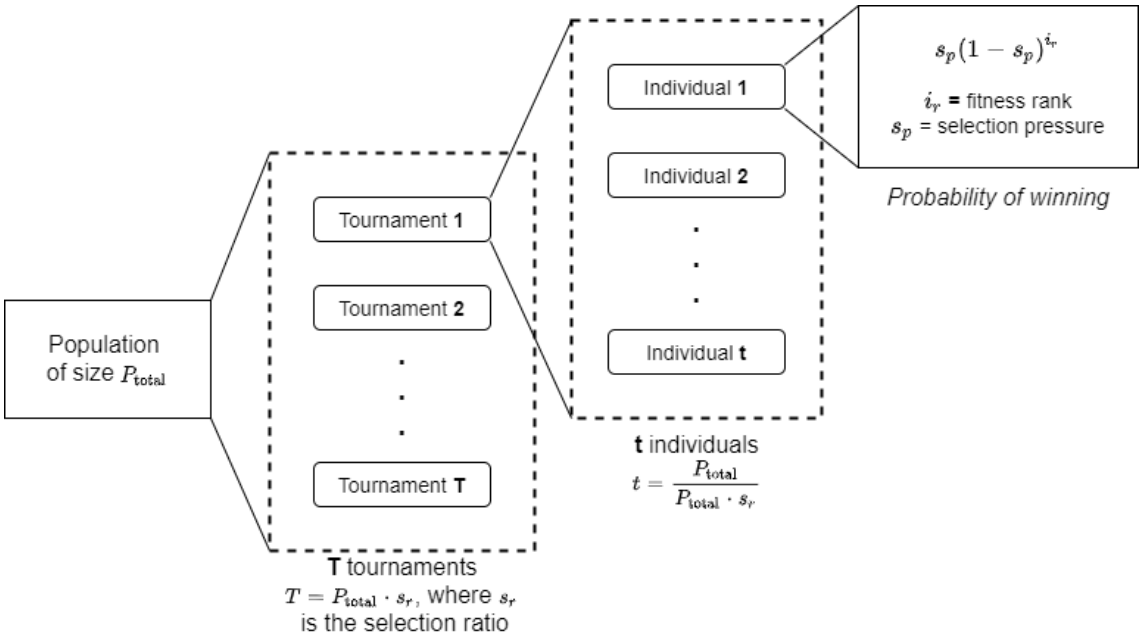


Figure 8: Structure of tournament selection applied to a population

A population of individuals is split into tournaments based on the selection ratio s_r, which defines what percentage of individuals will be selected, and the size of the total population P_{total}. From each tournament only one individual is selected with a probability defined by Equation 11 given below

$$s_p(1 - s_p)^{i_r}, \tag{11}$$

where s_p is the selection pressure and i_r is the rank of the individual in comparison to others in the tournament. Given for example a selection pressure s_p of *0.9* and a selection ratio s_r of 0.1 – meaning we select 10% of the initial population – the top individual in each tournament would have a 90% chance of being selected, the second best 9%, the third best 0.9%, etc. By finetuning the selection ratio and selection pressure, we can thus effectively decrease or increase the selective pressure applied to the population in question. In the case of speciation this selection process is applied to each species individually.

4.3.3 Fitness evaluation

In Equation 5 we presented a basic method for calculating the fitness of a solution. This equation can have significant problems with certain types of optimization situations and can be improved on.

One problem encountered when running the generic genetic algorithm, was an inability to converge to a viable solution when the operational area has multiple areas that have little or no relation to each other. This means that placing a beacon to provide localization for some area will not provide any benefits to other unconnected areas and will require a minimum number of beacons until any results are reached. Later, when defining the experimental setup, this will be presented in more detail. To improve convergence in this situation an additional metric called *localizability ratio* is presented in Equation 12

$$LR = Localizability\ ratio = \frac{\sum_1^m \max\left(1, \frac{b_p}{n}\right)}{m}, \tag{12}$$

where m is the number of points in the operational area, b_p is the number of beacons visible at some point p, and n is the minimum number of beacons required to provide localization. This provides a value ranging from 0 to 1 that represents how localizable the operational area is, and how close it is to achieving localizability. Placing beacons that provide coverage for an operational area that is not localizable yet will provide utility even though a minimum threshold for localizability has not been reached. Using this with the previously presented fitness equation gives us Equation 13 presented below

$$fitness = WS + LR - c \cdot b, \tag{13}$$

where WS is the well-seen ratio calculated from the geometry of the beacons, LR is the localizability ratio, c is the beacon cost, and b is the number of beacons used in the solution.

The value given to the beacon cost$p$ can have a large impact on the final result and the appropriate value can vary depending on the size of the operational area. The more beacons need to be placed for an “optimal” solution, the lower the beacon cost should be. This is because each individual beacon will provide a smaller increase in any optimality metric, since it can only reach a small portion of the entire area. Too high of a cost when the solution will require a large number of beacons to be placed can cause areas which will not be properly covered because the cost of coverage is too high.

To better deal with this, beacon cost will be automatically determined based on the operational area. Some assumptions are made that might reduce the validity of these

4.4 Problem specific genetic algorithm

The previously mentioned modifications to the genetic algorithm are presented below in Figure 9.

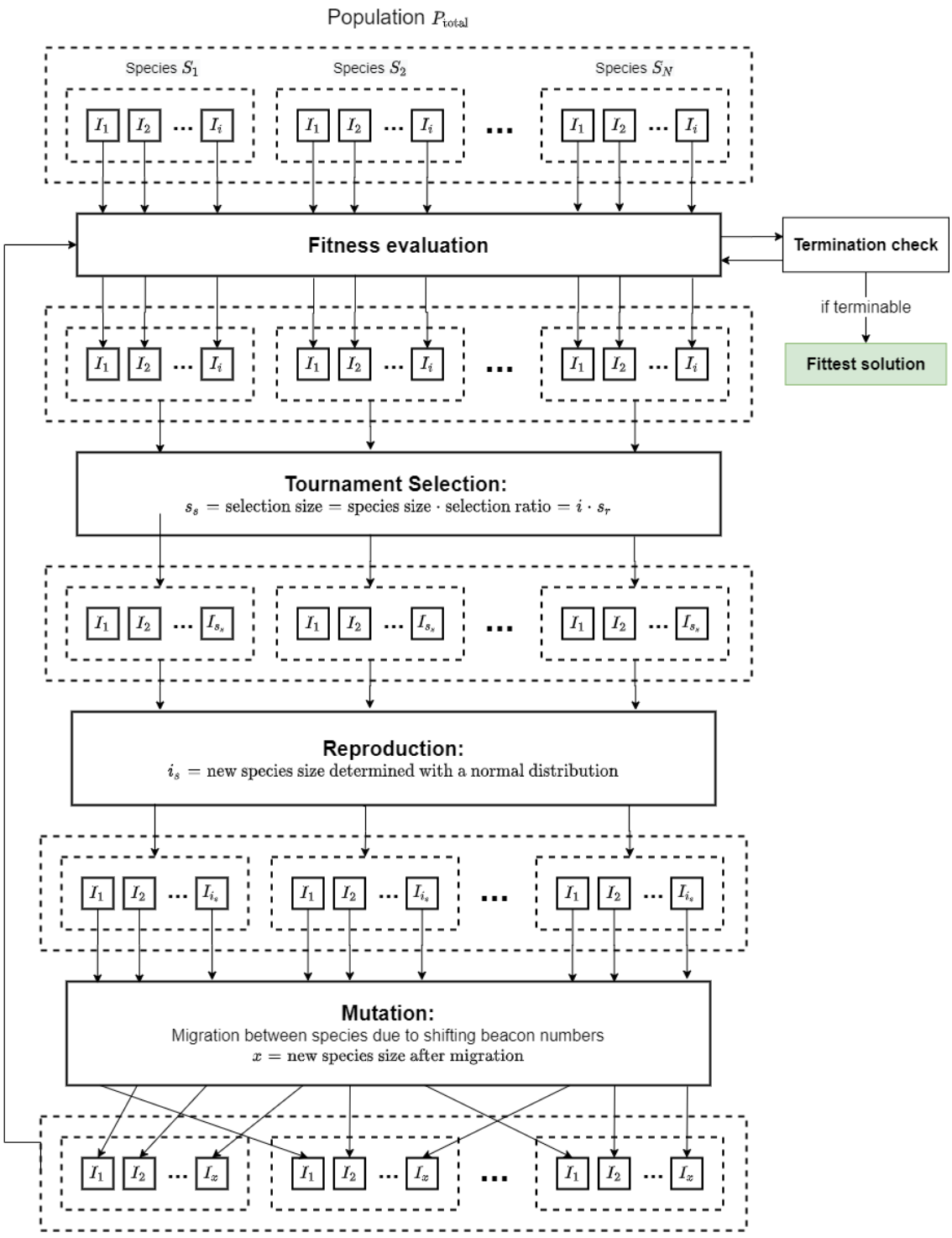


Figure 9: Structural overview of the problem specific genetic algorithm

As can be seen, the addition of speciation complicates the structure and operators may be applied to the population either individually or within a species. The specifics of each

block (fitness evaluation, tournament selection, reproduction, mutation) are detailed in their individual chapters. From the above figure you can get a general overview of how the population and species change in size as they pass through an iteration of the algorithm.

5. EXPERIMENTAL SETUP

The geometry of the optimisation environment can present a variety of difficulties for optimisation and the final result. We show some hand-made geometries, the problems they present for optimisation, and what kind of convergence on optimal solutions we expect.

Automatic and random generation of geometries can provide a wide range of different problematic optimisation geometries. The issue is that the geometries will need to be manually checked anyway to remove duplicates, trivial geometries, and those that are unsolvable. Instead, various geometries will be hand-made to represent a wide variety of optimisation problems due to obstacles and range limitations.

5.1 Single obstacle – centre placement

A single line-of-sight obstacle placed centrally in a rectangular area provides the most basic problem for beacon placement optimization. Figure 10 shows an example of convergence onto an optimal solution with several HDOP maps of various beacon placements.

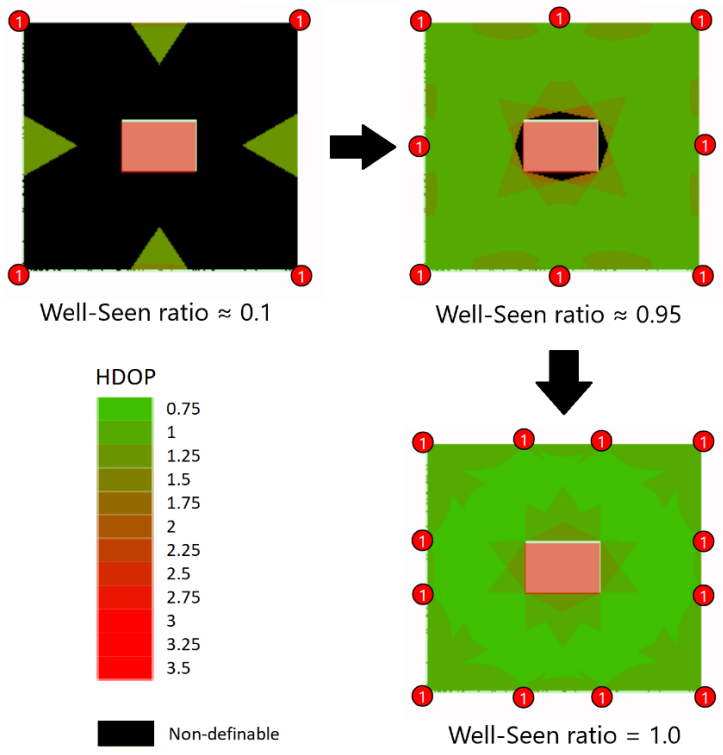


Figure 10: Beacon placement HDOP maps with a centrally placed LOS obstruction. Red circles are single beacons

Given a rectangular area and beacons with sufficient range, the simple and optimal solution is to place beacons in each corner. In the left-most solution, it can be seen how poorly this approach performs when there is a centrally placed obstacle. The coverage drops below the minimum of four beacons that is required for localisation in most of the operational area. Thus, HDOP is non-definable in most of the area, and the well-seen ratio is very low.

In the middle solution we have placed additional beacons between the corners, and this improves the performance significantly. Apart from the areas near the edges of the rectangle, the HDOP is definable for the entire area, but receives some poor values near the rectangle. The well-seen ratio supports this as ~5% of the area remains outside the well-seen coverage.

Finally, the right-most solution presents the intuitively optimal solution to this problem with regards to the well-seen ratio. As can be seen, issues with poor and non-definable HDOP have been eliminated, and the well-seen ratio has reached its maximum value. This is the solution we are expecting algorithms to reach for this specific map. When dealing with a single centrally placed obstacle we are expecting beacons to converge inline with the edges of the shape. The convergence points with different geometries are presented in Figure 11.

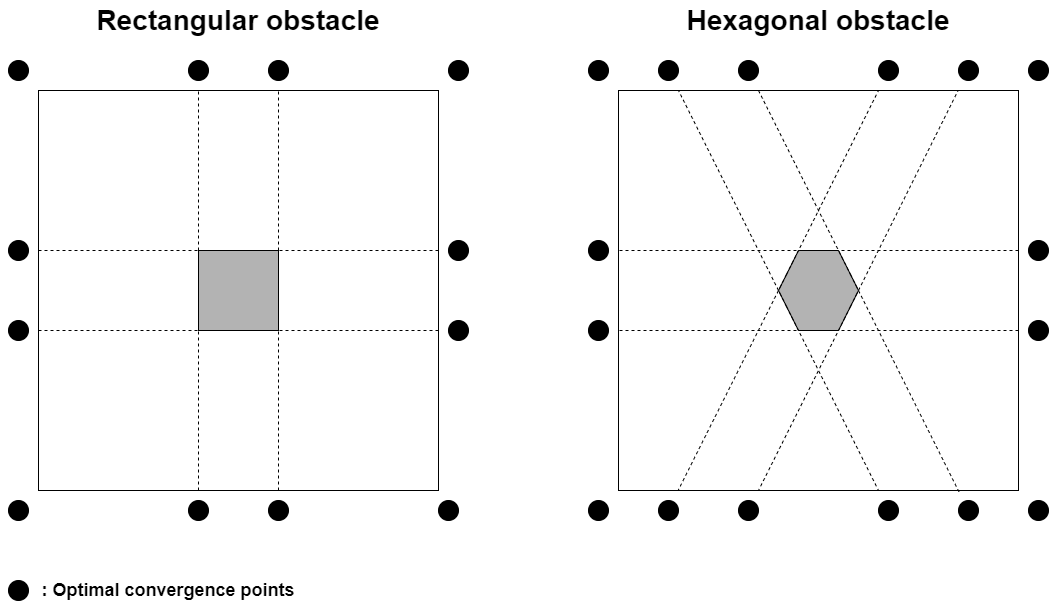


Figure 11: Expected convergence points with a rectangular and a hexagonal LOS obstruction. The obstacle is denoted with grey and the operational area as the white area.

For simplicity, we will be using a centrally placed rectangle for testing, but this rule double be applicable to obstacle geometries of varying and non-symmetrical shapes. These illustrations are generic, and do not account for the ranges of the beacons or the size of the area.

5.2 Single obstacle – corner placement

A corner placement of a single obstacle is a special case of free-form placement of an obstacle. The beacon convergence for a centrally placed obstacle, as shown in the previous chapter, would provide results just as well with regards to the HDOP and well-seen metrics, but solution optimality also factors in the number of beacons used. Convergence onto an optimal solution is presented in Figure 12.

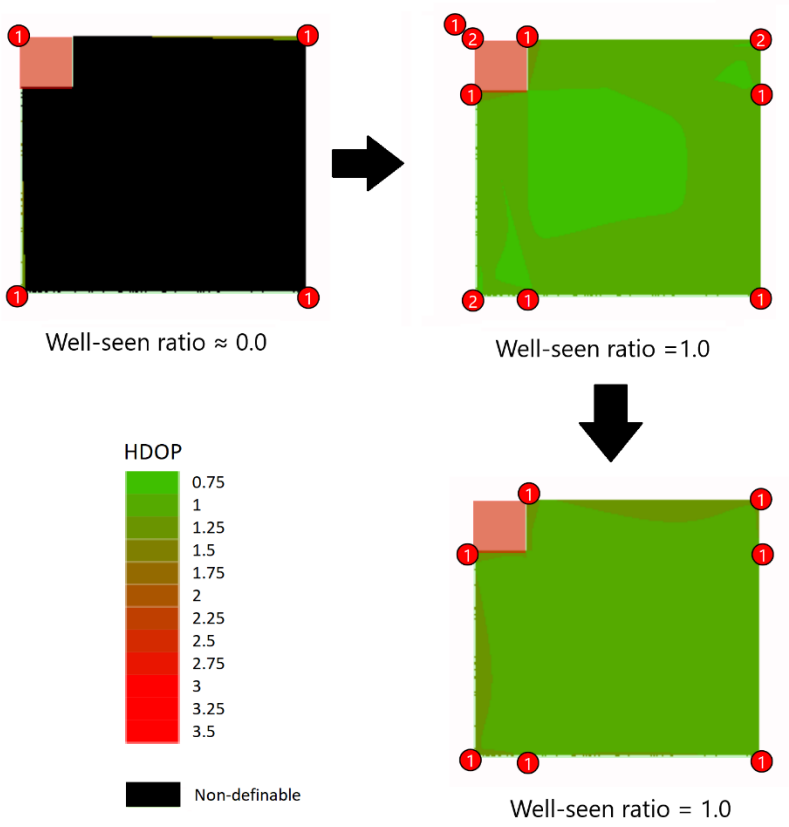


Figure 12: HDOP maps of different beacon placement with a corner placed LOS obstruction. Red circles are either single or double beacons (1,2).

Any beacons placed along the edges that are not facing operational areas do not provide any additional improvement for either HDOP or the well-seen metric. In the left-most solution with beacons placed in corners, the entire area has non-definable HDOP. The central solution has the beacons placed following the expected placements for a centrally

placed rectangular object. The HDOP and well-seen results are that of an optimal solution, but many of the beacons placed provide either marginal or no improvement with regards to these metrics.

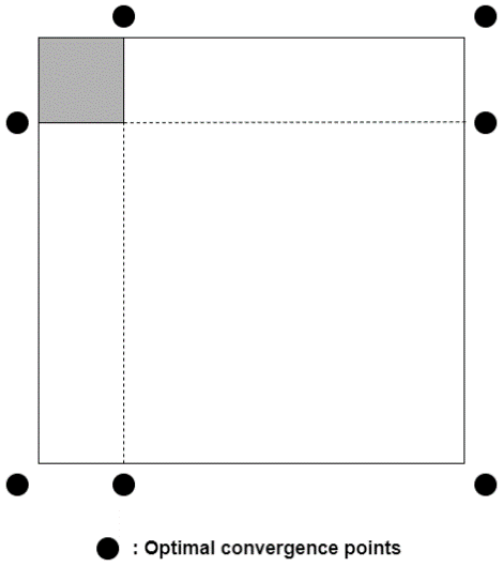


Figure 13: Expected convergence points with a rectangular corner placed LOS obstruction

The right-most solution presents the expected solution to this problem with regards to the well-seen ratio. Only 7 of the 12 beacons were required to reach an optimal well-seen ratio and while the HDOP does improve with more beacons, the change is often only marginally better. In Figure 13 is shown a visualization of how we expect the beacons to converge in a corner placement case, based on the previously shown convergence for objects shown in Figure 11.

5.3 Splitting obstacle

With the previous examples of centrally and corner placed obstacles, the convergence onto the expected points can happen quite gradually and even a single beacon placement can provide improved results. A more difficult case is when an obstacle splits the operational area into new sub-areas that do not rely on the same beacons for localization, or if they do, only marginally so.

This can cause major issues depending on the algorithm used, as the areas are essentially optimized independently of each other. Figure 14 shows an example of an area split into two by an obstacle, and what the HDOP maps look like with different solutions

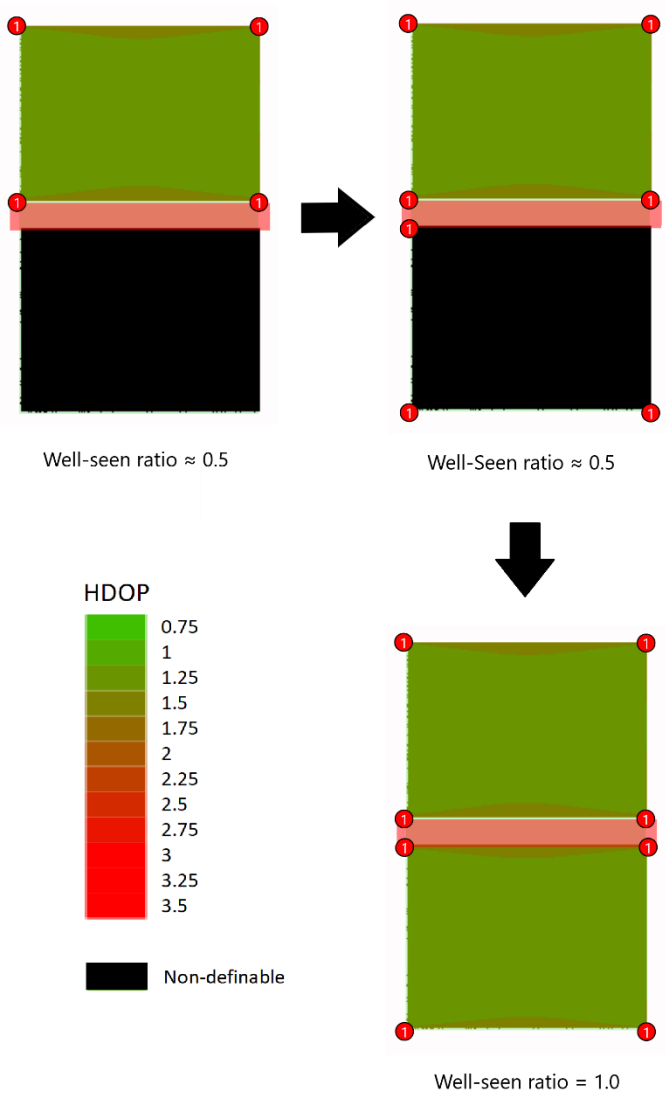


Figure 14: HDOP maps of different beacon placement with a splitting LOS obstruction. Red circles are single beacons.

In the left solution, the upper area has an optimal beacon placement, while the lower area has no coverage and a non-definable HDOP. Even when adding beacons to the lower area, as shown in the central image, the lower area still receives no coverage and has a non-definable HDOP. Only with the addition of the 4th beacon in the right image do these values improve. This is a difficult problem for an iterative algorithm, as the first 3 beacons, no matter how optimally they are placed, provide no coverage and thus no improvement in scoring. These beacons, however, have an associated cost with them, which can easily cause solutions attempting to optimize the lower area to be discarded before they provide any results.

5.4 Limited range

In all previous examples, the ranges of the beacons have been assumed to be sufficient to cover the entire area from all corners. When faced with an area where beacons will have insufficient range to cover it entirely, another set of problems for optimization come up. In Figure 15 is shown an example where the size of the area and the limited range of the beacons means that there is a very restrained way to optimally place the beacons.

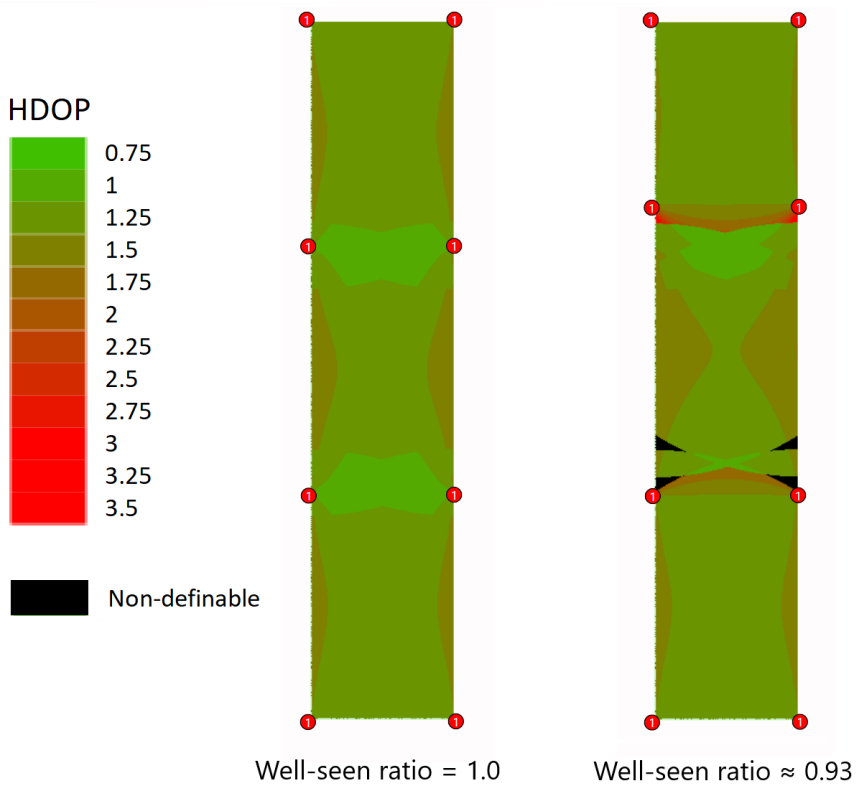


Figure 15: HDOP maps of different beacon placement with limited range. Red circles are single beacons.

In the left-hand side image, the beacons are placed so that they are just within range of each other to achieve an optimal well-seen ratio. The right-hand side image shows what even a small shift the in the beacon placements can cause. The well-seen ratio drops noticeably, and some areas lose coverage entirely. The size of this map and the ranges of the beacons are set so that it can only barely be covered by 8 beacons when they are placed optimally, with some wiggle room in the placement.

5.5 Multiple complicating factors

To provide a more challenging optimisation environment there will be a combination map that contains each of the previously presented problematic optimisation geometries. The map with a planned optimal solution is shown in Figure 16. As can be seen, there are

still some areas with poor HDOP. These are unavoidable in this optimisation configuration when placement of beacons is only allowed outside the operational area.

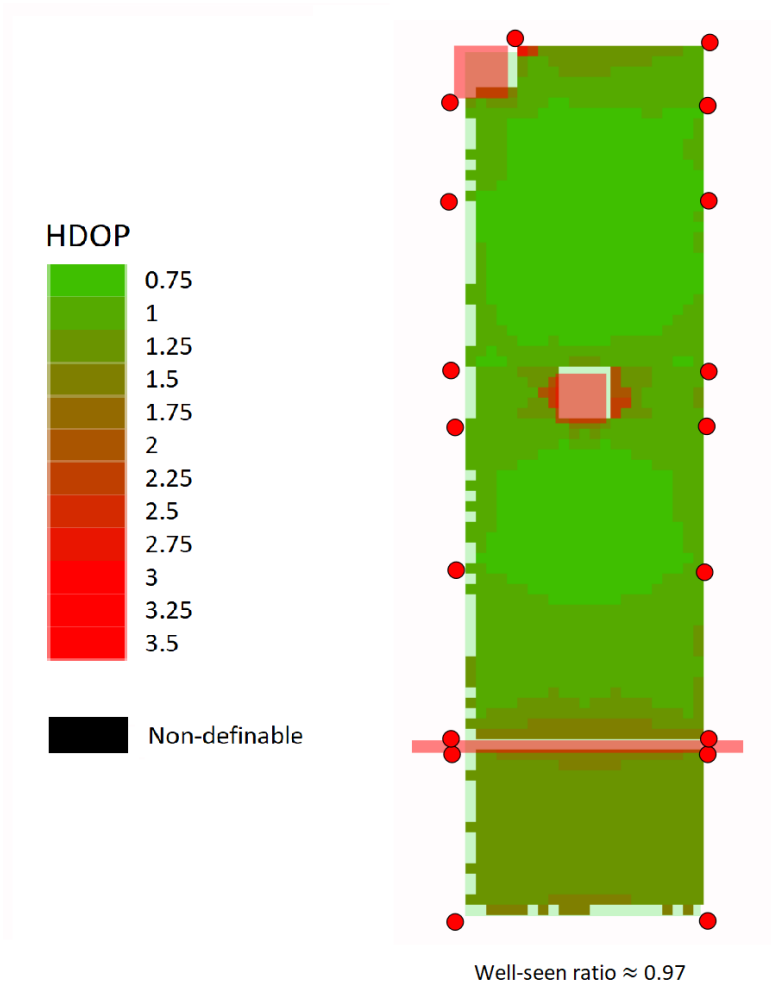


Figure 16: HDOP map of combined map. Red circles are single beacons

The purpose of this map is to provide a good range of problematic geometries and increase the number of beacons needed for an optimal solution. In the solution shown, we are expecting 18 beacons to be placed, with the placement following the rules described in the previous sections for each individual obstacle placement and beacon range-limitation. The solution shown is hand-made and thus might not be a truly optimal solution. Regardless, a well-seen ratio of about 0.97 is achieved, and this is expected. Some cannot achieve well-seen coverage since placement of beacons will be limited to the outside of the operational area during optimisation.

6. RESULTS

The performance of the problem-specific genetic algorithm will be compared with various parameters for species distribution. The problem-specific genetic algorithm, running with the best determined parameters, will then be compared with a generic genetic algorithm, running with configuration values that are otherwise identical. Both implementations of the genetic algorithm will be compared with other heuristic algorithms, such as hill-climbing and random generation.

6.1 Problem-specific genetic algorithm

The tinkering done with the generic genetic algorithm to improve its performance in this problem area added and changed the actions of various configuration values. One of these configuration values is the sigma value used to determine species distribution in speciation. The impact of this value on performance and the reasoning behind this should be determined, so that the strengths and weaknesses of this implementation of speciation can be identified. Below in table 2 is presented the fixed configuration values used for this comparison. The fixed configuration values used for this comparison are presented in Table 2

Table 2: Configuration values for sigma comparison

Population size	Species	Selection ratio	Selection pressure	Stop limit	Mutation rate	Translation range
2000	7	0.25	0.9	100	0.5	5.0

Figure 17 shows the performance of the algorithm with various sigma values. Note the cut-off of the y-axis at a well-seen value of 0.6. This is due to a very fast convergence to reasonable values within the first few generations, regardless of the sigma value.

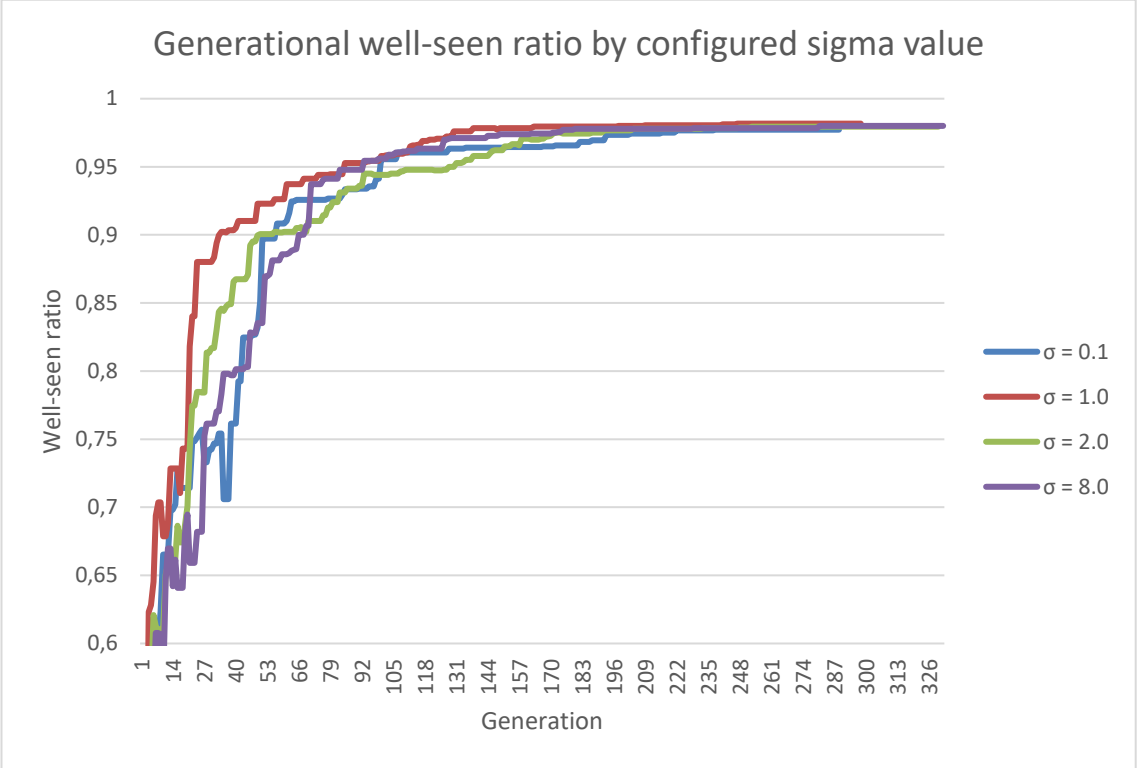


Figure 17: Well-seen ratio by configuration value for each generation.

The results have been collected from a single run with each sigma value. With randomized algorithms the results are unpredictable and can vary depending on the run. The processing time for the algorithm with the hardware available was poor enough that running it enough times to get a proper average would have taken too long within the scope of this thesis. These values may not be fully indicative of the performance but do seem to support what was noted during development and seem sensible when we later take a look at how the sigma value would affect the species distribution in each case.

What can be seen from Figure 17 is that, regardless of the sigma value, performance was quite good, with all of the configurations achieving a fast increase in fitness and eventual convergence onto > 0.97 well-seen ratio. The configuration of $\sigma = 1$ does seem to provide the fastest convergence to optimality as well as the best final result, while increasing or decreasing the value slightly decreases convergence speed as well as the final value. Of special note are the configurations $\sigma = 0.1$ and $\sigma = 8.0$, of which the former would cause limitation to a single species, and the latter would cause a completely flat distribution of individuals to all species.

With $\sigma = 0.1$ only the species within the average beacon number was allowed any individuals. Thus, this value in fact enforced a fixed beacon size for each generation based on the average number of beacons in the selected population. While at first it seems that

this could not provide any results, this still works since the average beacon number is determined only after the mutation and selection stages. If beacons need to be added for an optimal solution, there will be a large selection pressure towards the individuals that mutate more beacons during the selection round.

However, this is not a great method, since large populations of individuals will be lost before they are allowed to mutate towards optimality. Given a shift between generations to a different species being allocated all possible individuals, all other individuals will be lost. Due to the way selection applies pressure to the best individuals it might be that the population of individuals in the selected species is very small and much of the diversity present in the previous population will be discarded.

The configuration $\sigma = 8.0$ represents the opposite end of this issue. With the given configuration of the other parameters, this value provided a completely flat distribution, with each species allocated the same number of individuals. With the configured parameters in Table 2 each species was allocated 285 individuals, and this is a large enough population to begin to provide some results.

While the well-seen metric is used for fitness evaluation, and we can note that the $\sigma = 1.0$ configuration seems to provide the best convergence towards optimality as defined by the fitness, we cannot use it as a final comparison on localization performance. The well-seen metric is intended to broadly represent HDOP, but this is not assured, as it is a geometric simplification that does not account for spread of beacons in as HDOP does. Figure 18 presents the mean and standard deviation of the HDOP, as well as the validity ratio for the configuration $\sigma = 0.1$.

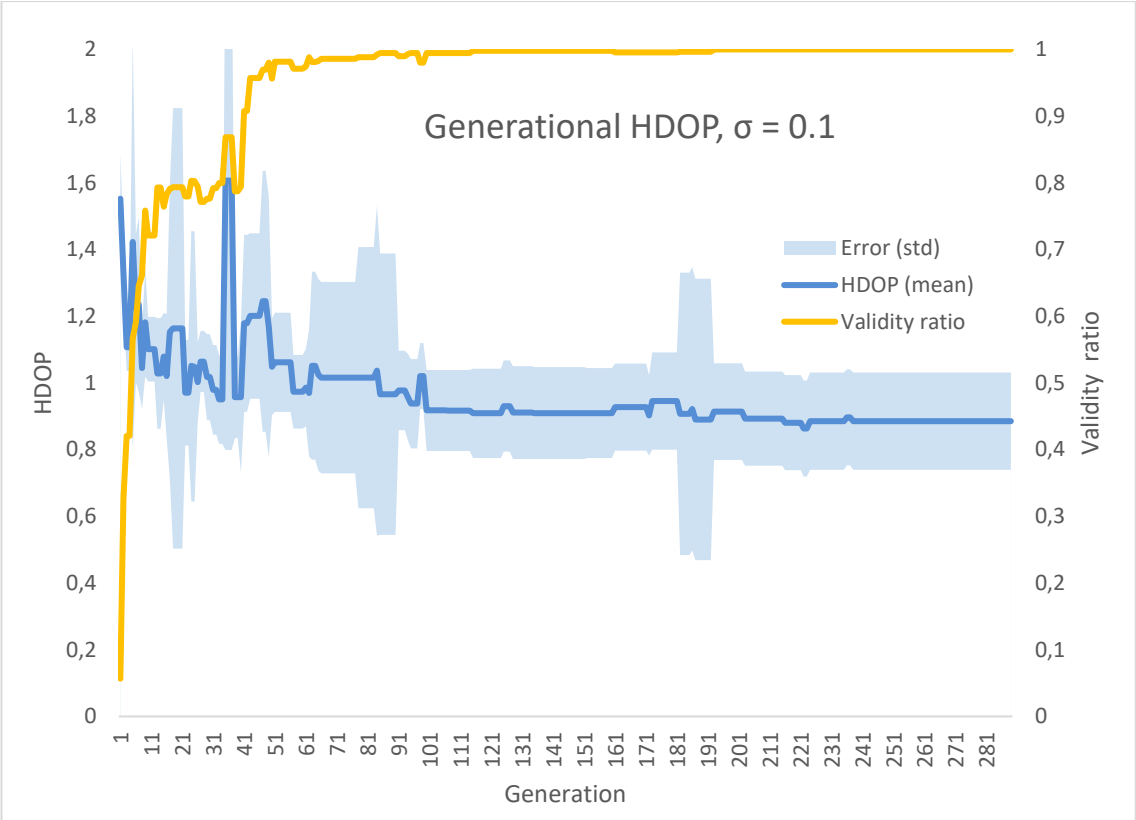


Figure 18: Mean HDOP with standard deviation, and validity ratio, by generation. σ = 0.1

As can be seen, this graph implies a connection between the HDOP and the well-seen ratio. Figure 17 previously showed that the well-seen ratio was being maximised during the optimisation process, and in Figure 18 we can now see that the HDOP was also being minimised. Even though we are not directly optimising for HDOP – as it is not used at any stage of fitness evaluation – we see it generally decreasing due to the connection between it and the well-seen metric. The validity ratio, as presented previously in Section 3.2.2, is the ratio of the HDOP points that can be used for evaluation. It is meant to show at what stage the HDOP is truly comparable with other results, as a low validity ratio means that a large portion of points are ignored in the mean and standard deviation calculations.

The results for the $\sigma = 0.1$ case in Figure 18 show a large variance in the HDOP and error values during the beginning of the optimisation process, as well as a low validity ratio. This is because at this stage there are various points that do not have sufficient coverage for localisation, and the points that do can have highly varying values, causing large errors in the mean HDOP. In Figure 19 the $\sigma = 1.0$ case is shown.

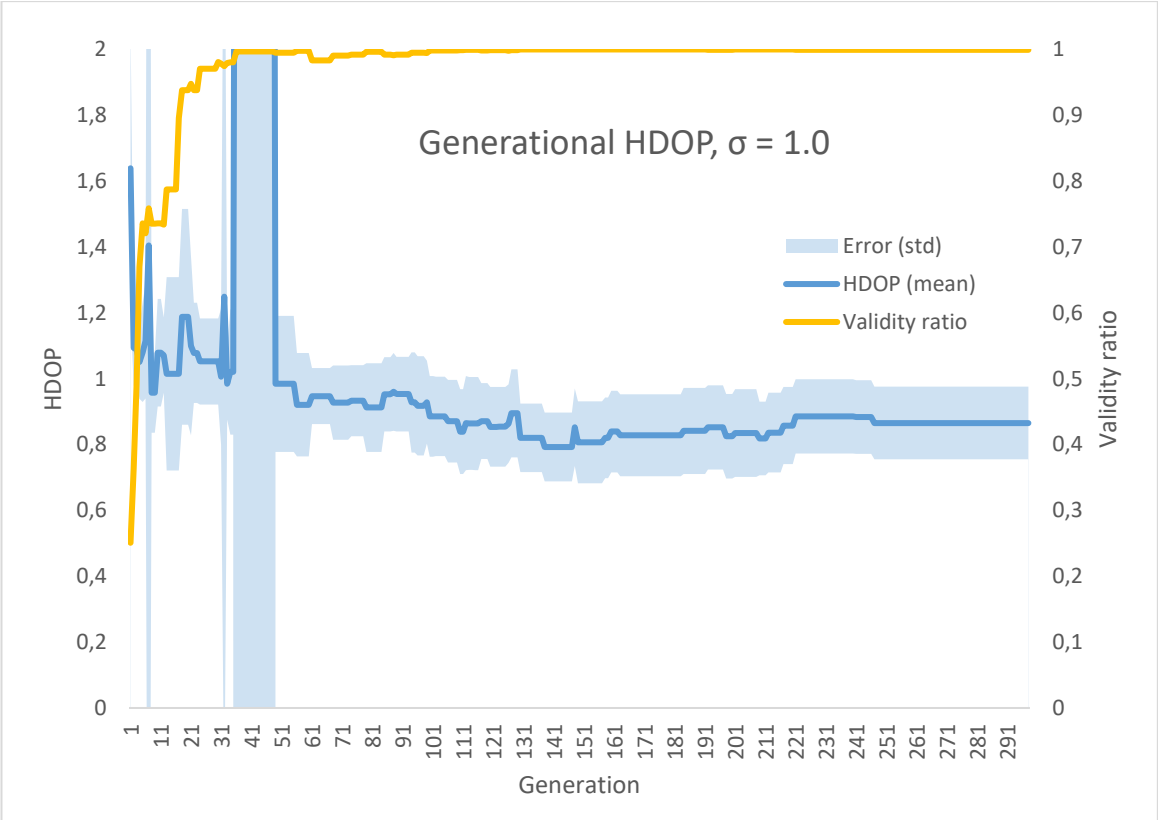


Figure 19: Mean HDOP with standard deviation, and validity ratio, by generation. σ = 1.0

There are a few important distinctions to note immediately in comparison to the previous case in Figure 18. Firstly, the validity ratio converges onto near total coverage of the operational area much faster than with the previous configuration. Secondly, the HDOP falls below a value of below 1 much faster than with the previous configuration. And thirdly, the error is less prone to spiking and remains consistent after the first ~50 generations.

While the slightly improved performance of the algorithm cannot be firmly tied to a better configuration value without more averaged trials, it seems to be supported by the other improved factors, such as a faster convergence of the validity ratio, faster convergence to a HDOP value below 1, and a generally smaller error during the process. All of these indicate better performance which could in addition allow better final convergence towards optimality.

Another thing to note, is that HDOP does not seem to be so strictly tied to our fitness value with the configuration $\sigma = 1.0$. From generation ~100 onwards the HDOP has not significantly improved and has instead even increased from the best results it managed during the run. This can be explained by looking at the number of beacons used by the

best individual in each generation. Below in Figure 20 is shown the number of beacons in each generation for various configurations of the sigma value.

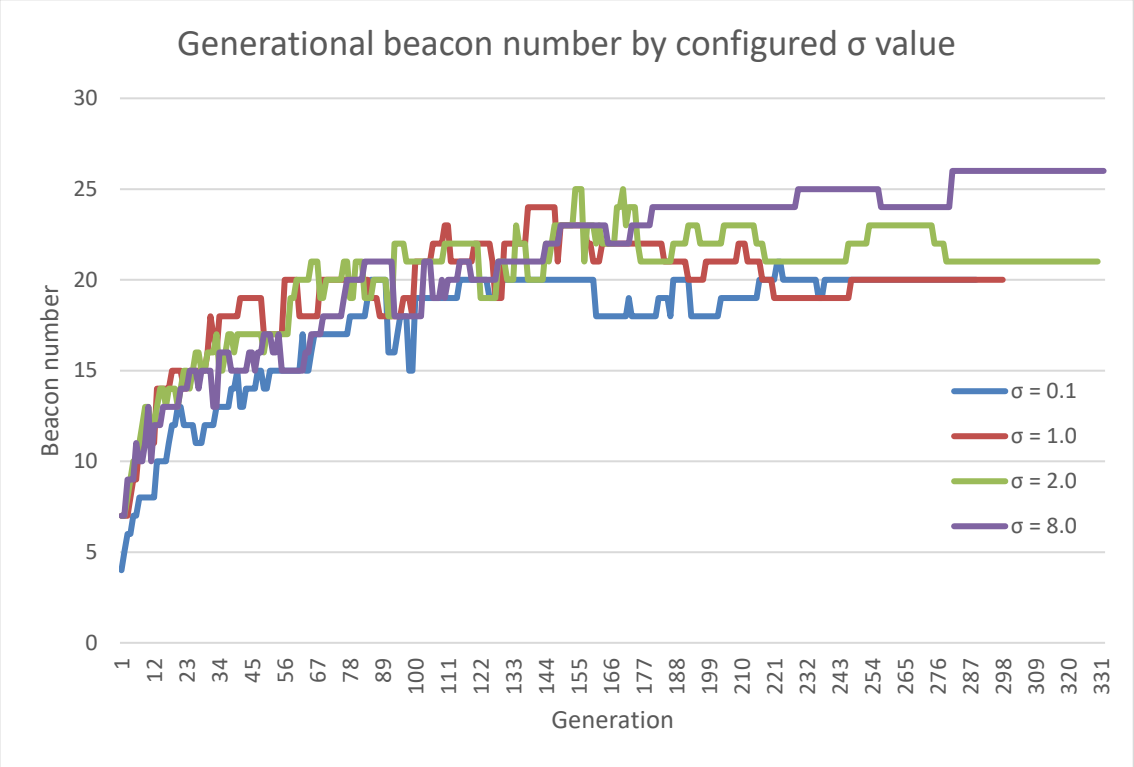


Figure 20: The number of beacons placed in each generation with different sigma values

As can be seen, the number of beacons used varied greatly between the algorithms run with different sigma values. We can now note why the HDOP value could end up increasing in general and why specifically with the configuration $\sigma = 1.0$.

The general reason for increasing HDOP during optimisation is simply caused by the factoring of additional aspects into our complete fitness value. Specifically, the addition of a beacon cost will cause minimisation of the number of beacons and will always cause beacons to be preferred to be removed if they do not improve the well-seen ratio. The well-seen ratio is not entirely connected to HDOP, and if points are already within the convex hull of their visible beacons, any additional beacons will not improve the well-seen ratio. With HDOP, however, each additional beacon will provide improvements dependent on its actual placement, with diminishing returns as more and more beacons are placed with a proper geometry. Due to these factors, when we minimise the number of beacons, HDOP is expected to increase some small amount, but as can be seen in Figure 19 the increases are very small and we are still dealing with HDOP values lower than 1.

The reason why this problem is exhibited more consistently in the case $\sigma = 1.0$ – and in addition the case $\sigma = 2.0$, which will be analysed next in more detail – is most probably due to the faster decrease of the HDOP to a nearly final value. Additionally, a larger amount of time is spent alternatively over placing beacons, and then subsequently removing those that have no impact on the fitness. This will cause much more shifts in the HDOP value than with the other configurations that do not exhibit the same over-placement and then removal behaviour. Now, let us take a closer look at the configuration $\sigma = 2.0$ shown in Figure 21.

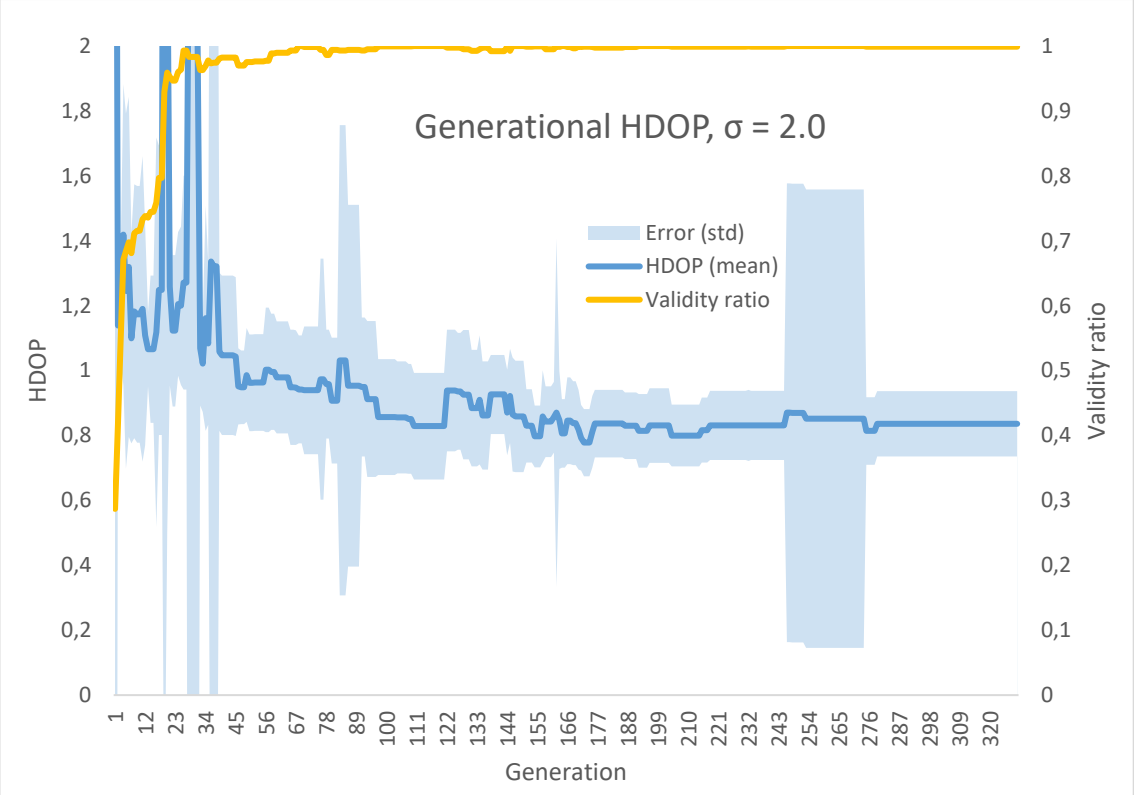


Figure 21: Mean HDOP with standard deviation, and validity ratio, by generation. σ = 2.0

This configuration is more comparable in performance to the configuration $\sigma = 1.0$. In comparison to the configuration $\sigma = 0.1$ shown in Figure 18, it has faster convergence of the validity ratio to near total coverage of the operational area, faster convergence of HDOP to below 1, but a similar number and intensity of spikes in the error graph. Other than the increase in the error spikes in comparison to $\sigma = 1.0$, this seems like expected behaviour.

When taking a look at the number of beacons placed between $\sigma = 1.0$ as shown in Figure 19, and $\sigma = 2.0$, as shown in Figure 21, we can note some differences between the two configurations. While both configurations exhibit the behaviour of over-placing beacons and then removing them, the configuration $\sigma = 1.0$ seems to be less inclined to remove

beacons and generally has a larger number of beacons in its solutions. The configuration $\sigma = 2.0$ on the other hand consistently has a lower number of beacons.

Now let us take a look at our final configuration of $\sigma = 8.0$ shown below in Figure 22.

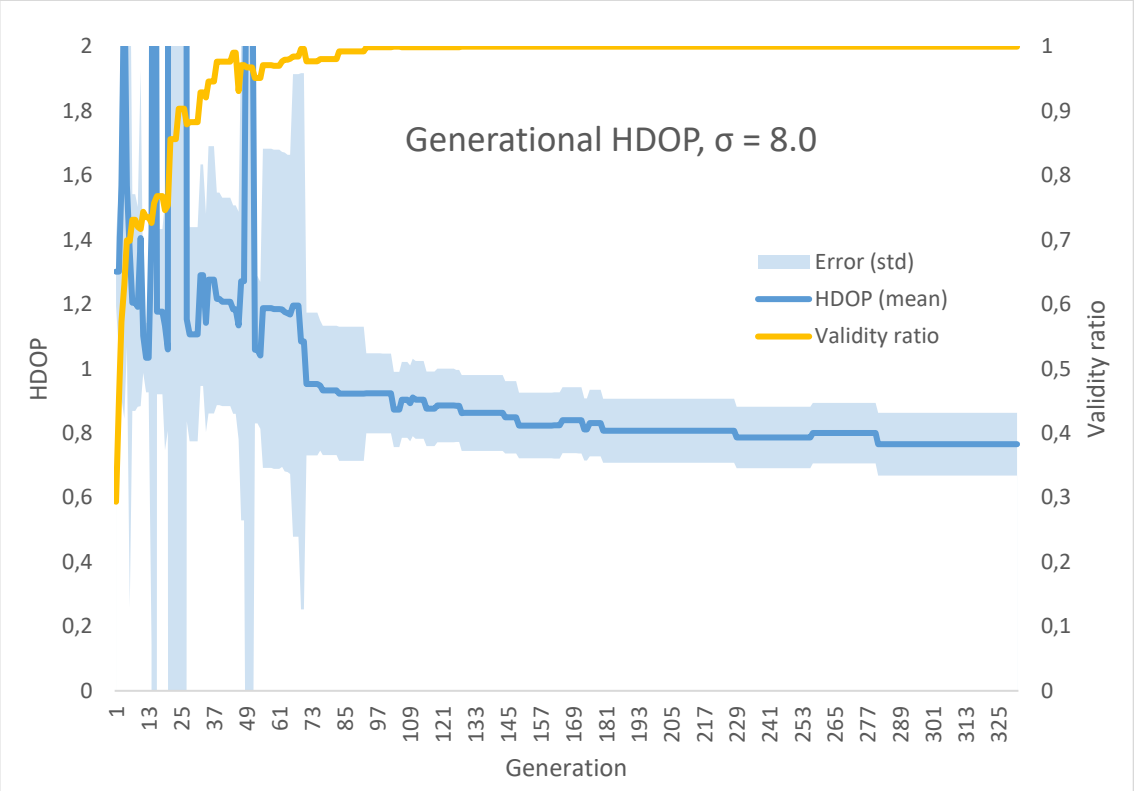


Figure 22: Mean HDOP with standard deviation, and validity ratio, by generation. σ = 8.0

As explained previously, the configuration $\sigma = 8.0$ is a large enough value to cause a totally flat distribution of species sizes. With a quick look at the HDOP graph it would seem that this configuration provided quite comparable results to the previously presented configurations.

The validity ratio does not converge to near total coverage of the operational area quite as fast as with the other configurations, but does provide improved results compared to the configuration presented in Figure 19. The HDOP value actually decreases below 0.8 and provides better results than any of the previous configurations. And finally, the error seems to be focused on the beginning of the process, and no additional spikes are seen. However, when a closer look is taken at the number of beacons placed in Figure 20 these results become clearer.

With a configuration of $\sigma = 8.0$ there is a large overplacement of beacons. Unlike with the lower configurations of $\sigma = 1.0$ and $\sigma = 2.0$, there is not as consistent culling of bea-

cons after overplacement, which cause the final solution to have large number of beacons. As previously noted in Section 5.5, the map used for optimisation should have an optimal solution of 18 beacons, and the final solution with this configuration ended up at 26 beacons. This is far from the optimal number, and far from the other solutions provided by runs with other configurations. This explains why this configuration exhibited the best HDOP result, while simultaneously providing a well-seen ratio similar to $\sigma = 0.1$ and $\sigma = 2.0$ configurations. Each additional beacon will almost always increase the HDOP slightly, but diminishing returns are apparent as more and more are placed. In comparison to the configuration $\sigma = 1.0$, the placement of 7 additional beacons only increase the HDOP by < 0.1 which, while not completely insignificant, is not worth the placement of extra beacons as any HDOP under 1.0 is already a good result.

An interesting note about the well-seen ratio achieved with the configuration of $\sigma = 1.0$ is that it is in fact better than the optimal value that was predicted during introduction of the combined map in Section 5.5. The solution for it was designed by hand with the idea that the resolution of the coverage map would be large enough to avoid any discrepancies. The most probable reason for the larger than expected final value is the usage of quite a low resolution during optimisation due to memory constraints. The problem a low resolution may present is shown in Figure 23.

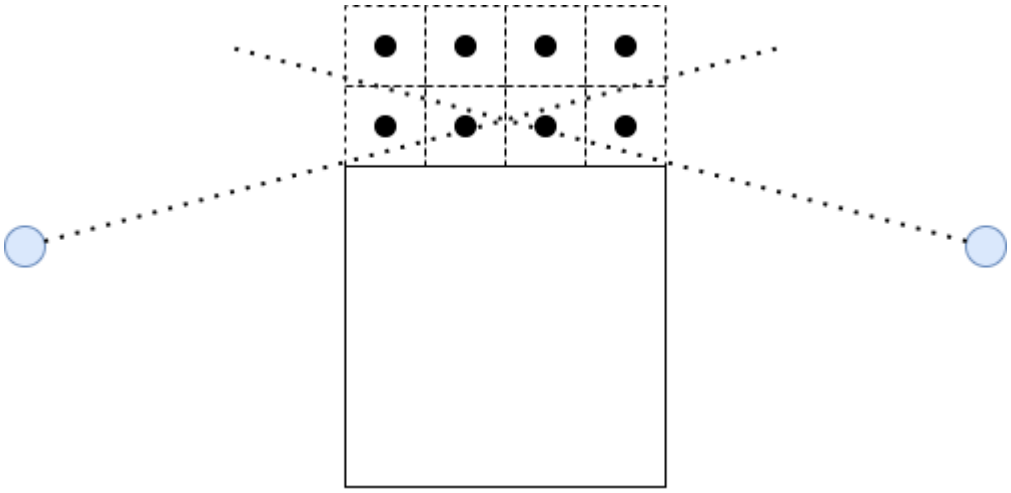


Figure 23: Low discretization resolution impacting coverage

The black dots represent the points in the coverage map based on the discretization grid. In this example, all points are visible to both beacons marked in blue, even with the obstacle cutting off the view partially. However, if the resolution were increased, the area below the lines and above the obstacle would have new points that would not have visibility to the beacons with the given positioning.

To conclude, while all configurations of sigma values provided good results with high well-seen ratios (>97%) and low HDOP values (<1.0), the best results were with the configuration of $\sigma = 1.0$. With this configuration, a well-seen ratio of ~98% and a median HDOP value of ~0.86 was reached while only placing 20 beacons. This is close to the optimal solution that the map was designed with, and all later examples of the problem-specific genetic algorithm will be using this configuration of $\sigma = 1.0$.

6.2 Performance comparison

One of the major goals of this work was to analyse how a genetic algorithm can be tailored for the specific problem area of beacon placement for localization purposes. We will be taking a closer look and comparing its performance with various other algorithms, such as a generic variant of the genetic algorithm, a hill-climbing algorithm, and a random generation algorithm.

6.2.1 Genetic algorithms

First, let us look at how the performance of the problem-specific genetic algorithm compares to the generic implementation. The generic implementation is missing implementation for localizability in the fitness evaluation, a more complex selection process, and speciation. Figure 24 shows the performance of a re-run of the problem-specific genetic algorithm

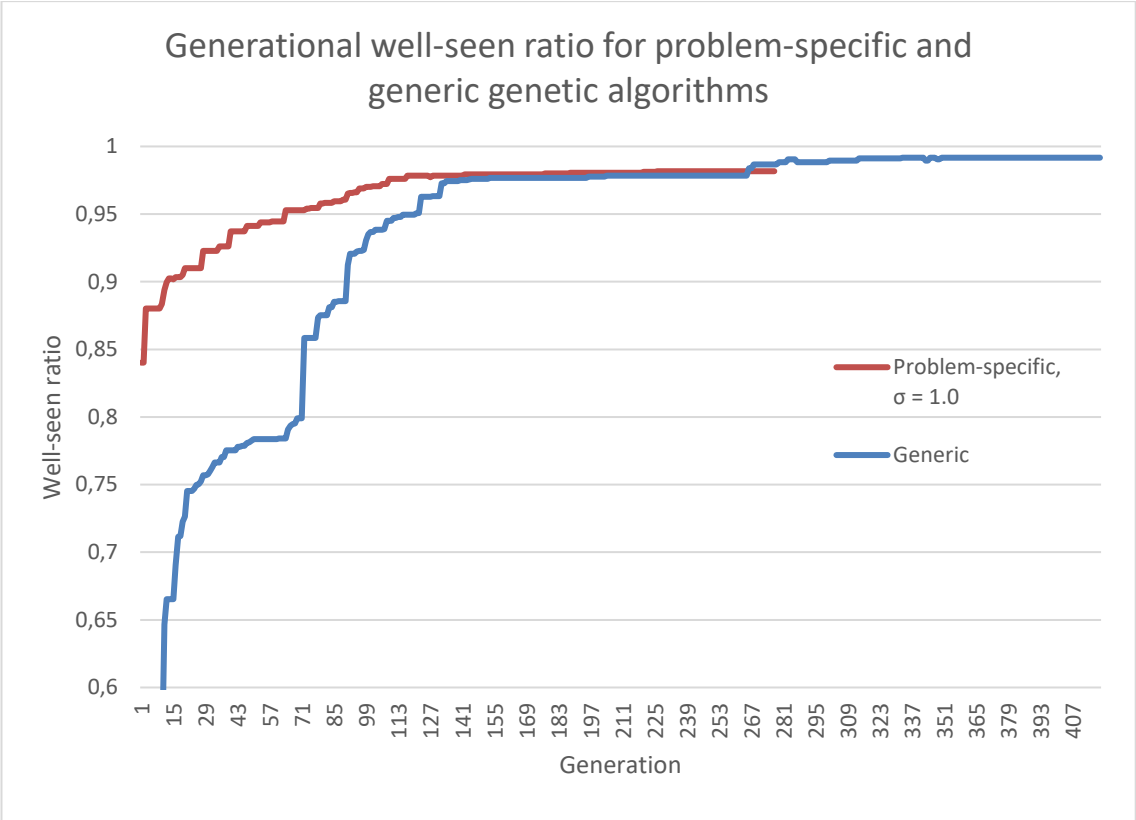


Figure 24: Well-seen ratio values for problem-specific and generic genetic algorithms for each generation.

The end results are not quite as clear as was expected during initial development. The generic implementation in fact provides better results when comparing the final well-seen values but does converge significantly slower in the earlier generations. The problem-specific algorithm reached a well-seen ratio of ~0.98 while the generic implementation reached an improved result of ~0.99. The difference is not particularly large but does indicate that the generic algorithm does have better convergence. Now, let us take a closer look at the generational HDOP values for both algorithms in Figure 25 and Figure 26

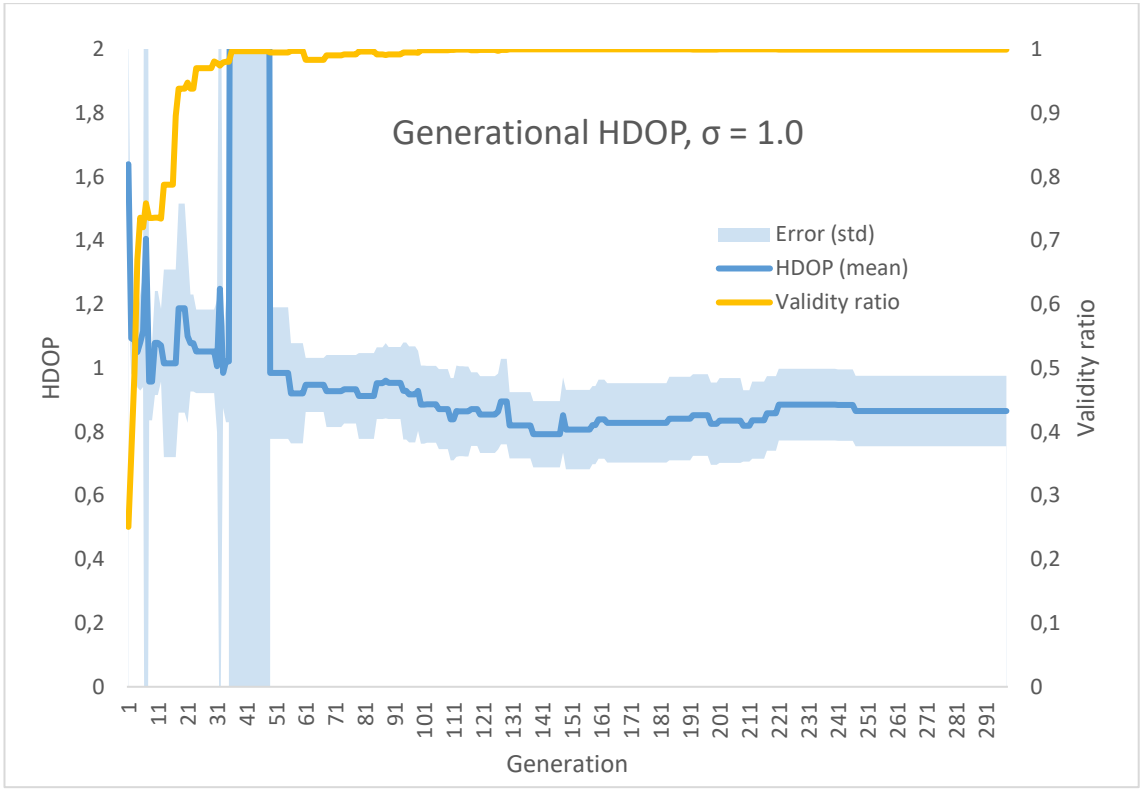


Figure 25: Mean HDOP with standard deviation, and validity ratio, by generation. σ=1.0

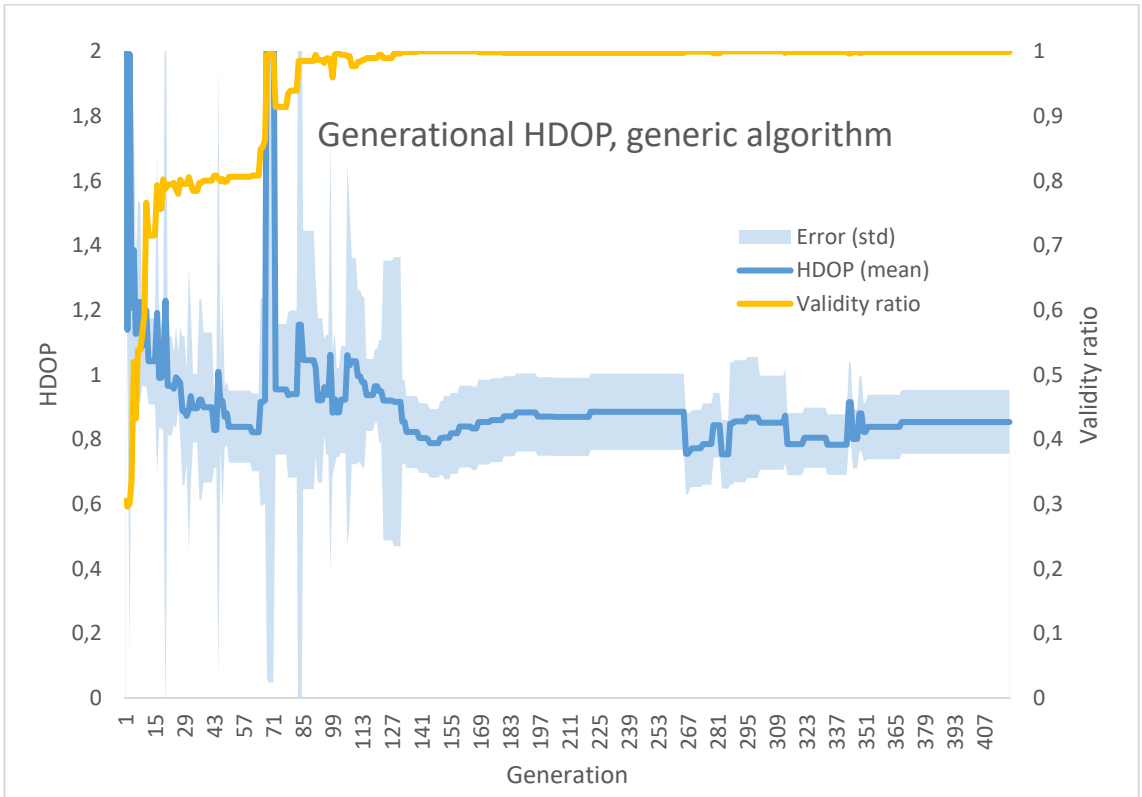


Figure 26: Mean HDOP with standard deviation, and validity ratio, by generation. Generic genetic algorithm.

Both implementations converged on low HDOP values (<1.0) with the generic implementation having a slight advantage with a value of ~0.85 in comparison to ~0.86 for the problem-specific implementation. However, the rate of convergence to a high enough validity ratio was poorer. A likely explanation for the slower convergence towards having all points with definable HDOP values is the problem of the splitting obstacle in the combined map. Due to not using the localizability ratio in fitness determination, the algorithm had trouble placing beacons in the lower area as shown previously in Figure 16. The validity ratio was then achieved very suddenly when a solution was found that provided localisation in the lower area.

Another interesting note is the larger variability and spiking of the error in the earlier generations with the generic implementation. It is hard to exactly say what could have caused this, but the lack of strict limitation of the population to specific beacon numbers would be a possibility. Let us take a closer look at the number of beacons in Figure 27.

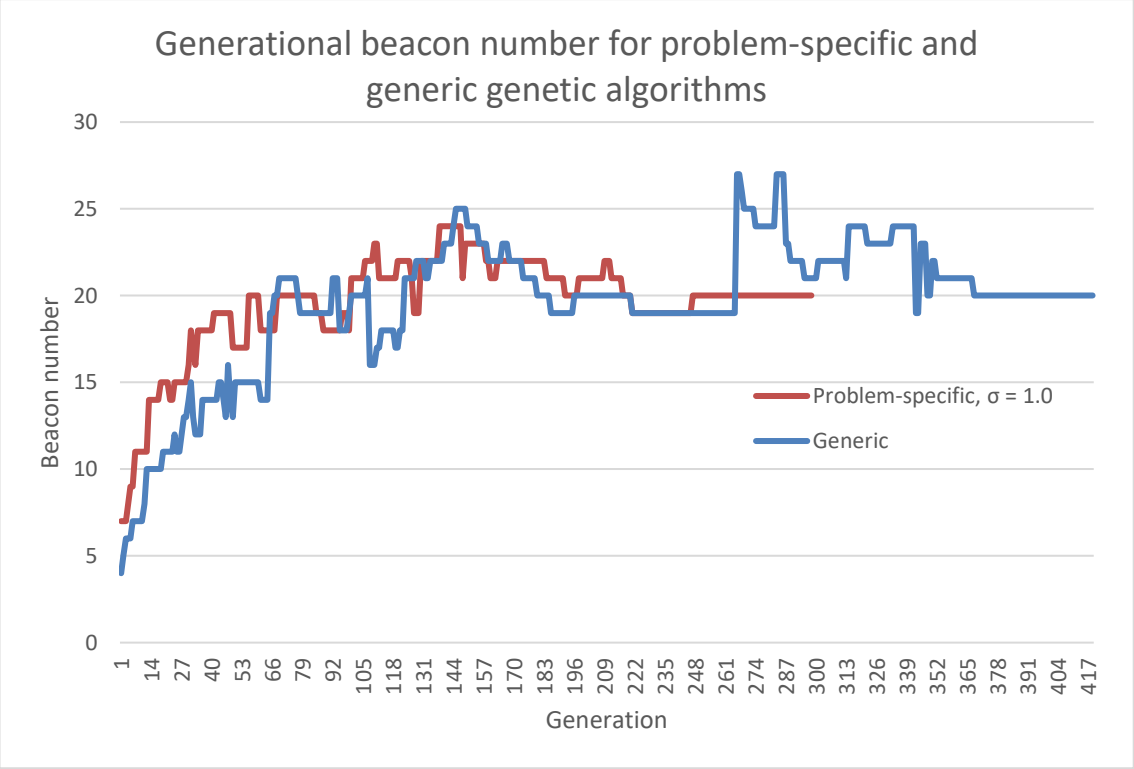


Figure 27: Beacon numbers values for problem-specific and generic genetic algorithms for each generation

The number of beacons used was very similar between both implementations until after ~250 generations, with only a faster increase in beacon number being of note for the problem-specific implementation. This would also indicate why the convergence to a large well-seen ratio was significantly faster. While the problem-specific implementation was finished after about 300 generations, the generic implementation took significantly

longer. This seems to be due to the generous over-placement of beacons at generation 267. The beacon number went from 19 all the way to 27, and this type of behaviour is not possible for the problem-specific implementation. This is because the number of beacons allowed in each generation is strictly limited around the average number of beacons in the selected population. If the average at that stage was around the best individual of 20 beacons, the maximum number of beacons for any individual would have been limited to 23 beacons with the species number of 7 that was used during optimisation.

Over-placement of beacons is generally a good thing, even when it is done suddenly within a few generations, and with large amounts of beacons placed. As long as any beacons that provide minor or no benefits to fitness are then removed, this is not problematic. This indicates the heavily constrained speciation implemented for the problem-specific algorithm is too limiting. It needs to allow a wider range of species, either through increasing the total population size and the species number, or completely removing the limitations on the number of beacons allowed, as is done in the generic version.

One way to improve the problem-specific variant would be to remove the strict limitation of crossbreeding between different species – as is done in the generic version – and instead, discourage crossbreeding with some other method. This could, for example, be a penalty applied during reproduction the farther away the individuals beacon numbers are to each other. This would implement speciation less strictly and allow the population to optimise across the entire search-space with less restriction, instead of being forced into some local area.

6.2.2 Other algorithms

If the genetic algorithm does not provide significant improvements over simpler algorithms, it is of limited use due to increased complexity and computation time. So, let us take a look at some simpler heuristic algorithms to get a better understanding of how the genetic algorithm performs in relation to them.

One of the simplest algorithms for solving this problem and many others is random generation. It simply involves randomly generating a solution and checking for any possible improvements to the well-seen ratio. One thing to note about this method is the requirement of a beacon number as input. This is because unlike algorithms which can gradually converge on the expected number of beacons, random generation must start from scratch each iteration and this must be done with knowledge of the expected number of beacons.

An implementation could be made that would allow an unknown number of beacons, but this would need additional mechanisms to increase or decrease the number as is deemed necessary. With a genetic algorithm this natural increase or decrease in the number of beacons happens due to a large number of solutions being iterated simultaneously. For random generation, with a single solution generated each iteration, the choice of whether to increase or decrease the number of beacons for the next iteration is not a trivial decision.

The second algorithm is a simple implementation of a hill-climbing algorithm, following the implementation by Allen et al. [1]. It involves making incremental changes to a solution, to slowly converge on the optimal result, as well as starting over with a fresh solution when certain parameters are met, to avoid getting stuck in local minima. As the previously mentioned random generation algorithm, this also requires a beacon number to be known beforehand. The optimisation results for this, along with both the generic and problem-specific genetic algorithm, as well as the random generation are shown below in Figure 28.

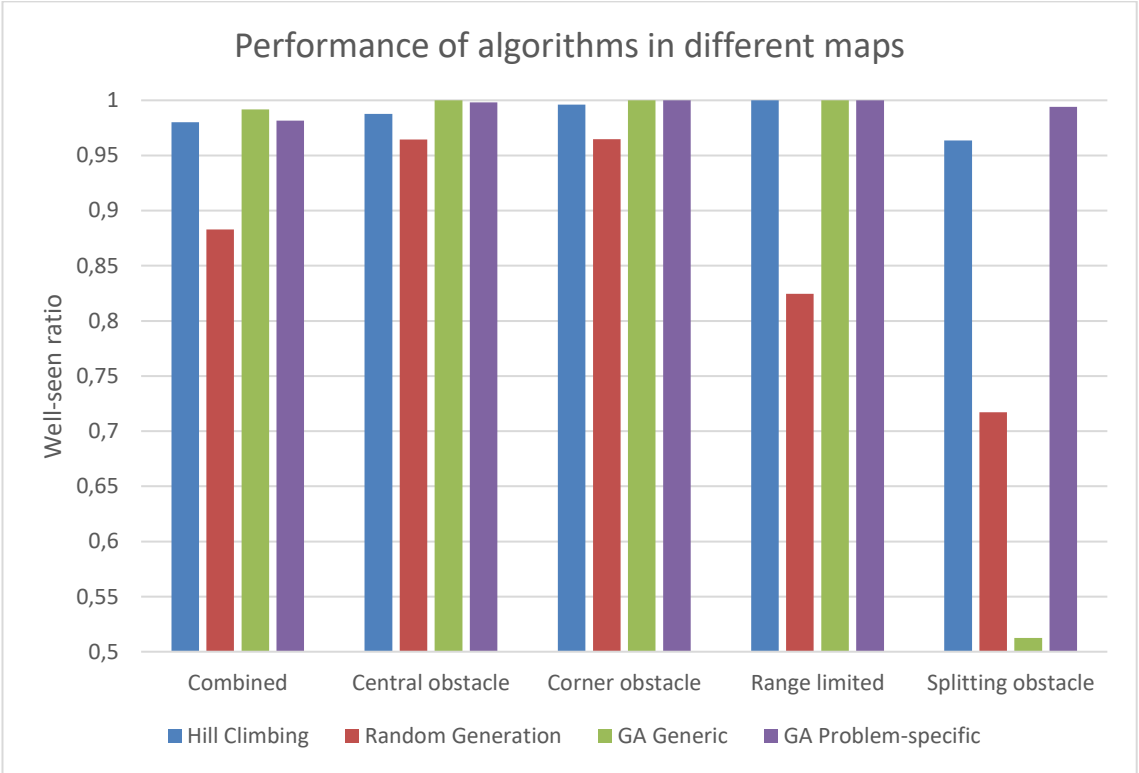


Figure 28: Algorithm performance in combined, central obstacle, corner obstacle, range limited and splitting obstacle maps

Both versions of the genetic algorithm provided the best results in most maps with significant improvements compared to the random-generation algorithm and less significant improvements compared to the hill-climbing algorithm. One interesting thing to note is

that the generic genetic algorithm failed to provide results in on of the sides in the splitting obstacle map but managed to provide results in the split area in the combined map. The lower number of beacons required for an optimal solution most probably prevented individuals from developing towards it as the beacon number would significantly increase from 4 to 8 beacons. In the combined map however, the number of beacons is much larger throughout the optimisation process, which means that placing beacons is less costly. Additionally, there are significantly more beacons during the optimisation process which do not provide any utility. Thus, it is more probable that there will be some individual which provides a good enough result otherwise, while still having beacons placed to provide coverage in the separate area that does not currently provide any utility. Such individuals would not be removed from the population so easily and would be able to progress into solutions that do eventually provide utility.

It would be interesting to test these algorithms with an even larger map to see if any differences arise between the hill-climbing algorithm and the genetic algorithms. However, memory limitations heavily constrained maps that required too many beacons, as memory usage directly scaled with the number of beacons placed by the entire population. The memory required to store each individual increases as the number of beacons in its solution increases. This means that as the average number of beacons in the population increases, so will the memory usage.

7. CONCLUSIONS

Beacon placement in navigation systems can have a large impact on the quality of localisation. This thesis took a closer look at the usage of genetic algorithms for placement design, with analysis of the performance of some problem-specific modifications, as well as comparison with other heuristic algorithms. The well-seen metric was used as the key metric during optimisation.

7.1 Conclusion

One of the key goals for this thesis was to evaluate the usage of the well-seen metric in optimisation. Its usage as an indicator of optimality in various algorithms, and how it related to the commonly used metric HDOP was evaluated. It was thought to mirror HDOP while being less computationally intensive to calculate, and this was evident in the results. While HDOP was not used by any of the algorithms during optimisation, the results obtained during each generation of the genetic algorithm were quite consistently showing a decrease in the average HDOP value of the operational area. There was some small variance where HDOP increased during optimisation, but this was often tied to the reduction of beacons. Placement of beacons that provide more coverage will always improve the HDOP result, but tend to provide diminishing returns with over-placement. There is a clear connection between the well-seen metric and HDOP.

Another goal was to evaluate the usage of genetic algorithms for beacon placement design. Some problem-specific modifications were made based on existing research in similar problem areas. One of these was the usage of speciation. The usage of normal distribution and how the spread of individuals into different species impacted the optimisation process was evaluated. The modified genetic algorithm was then compared with a generic variant in a wide variety of different optimisation scenarios designed to provide different challenges for optimisation. The results obtained were not quite what was expected during development of the genetic algorithm, as the generic variant ended up performing slightly better on some accounts than the problem-specific one. This was most probably due to the overly strict limitation of individuals to a specific number of species, instead of allowing completely free speciation with some penalty applied to limit cross-species reproduction.

In comparison to some other heuristic algorithms, the genetic algorithms provided better optimisation results as well as not requiring the number of beacons to be placed as a starting point.

7.2 Open research direction

Some questions came up during this thesis and remained unresolved. One interesting avenue of research would be to further evaluate how the well-seen ratio matches with HDOP. For example, with very narrow operational areas, the well-seen ratio might not match up as closely with HDOP, and similar well-seen values might provide drastically different results when comparing HDOP. Random generation of scenarios and further evaluation of those deemed problematic could expose some additional scenarios that were not predicted in this thesis.

The limitations to map size due to memory constraints prevented the usage of larger maps to compare the genetic algorithm with other algorithms. It would be interesting to see how well the results scale with an increasing number of beacons required to reach decent results, but this would require an improvement in the implementation to reduce memory usage.

Speciation in this paper was implemented with a strict limitation to certain species around the average number of beacons in the selected population and removing individuals not in these species. This could be too strict to allow proper development of individuals and it would be interesting to see how the performance could be improved with an alternate implementation. One alternative to the strict speciation would be to not completely prevent cross-species reproduction, but instead discourage it by applying a penalty in the reproduction stage between different species. Another possibility might be to use a different way to group individuals into species, since the beacon number is not necessarily the best way to determine how similar solutions are. This could, for example, be based on grouping individuals with similar beacon placements or perhaps those with similar fitness values.

REFERENCES

[1] The Range Beacon Placement Problem for Robot Navigation, in: CCCRV, IEEE, 2014, pp. 151-158.

[2] Beacon placement for range-based indoor localization, IEEE, 2016, pp. 1-8.

[3] Dilution of Precision, GPS World, Vol.10, 1999, pp. 52-59.

[4] Optimization of wireless locating in complex environments by placement of anchor nodes with evolutionary algorithms, IEEE, 2013, pp. 1-6.

[5] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M.B. Srivastava, Coverage Problems in Wireless Ad-hoc Sensor Networks, 2001.

[6] N. Bulusu, J. Heidemann, D. Estrin, Adaptive Beacon Placement, 2001.

[7] D. Moreno-Salinas, A.M. Pascoal, J. Aranda, Optimal sensor placement for multiple target positioning with range-only measurements in two-dimensional scenarios, Sensors (Basel, Switzerland), Vol. 13, Iss. 8, 2013, pp. 10674.

[8] Z. Fei, B. Li, S. Yang, C. Xing, H. Chen, L. Hanzo, A Survey of Multi-Objective Optimization in Wireless Sensor Networks: Metrics, Algorithms, and Open Problems, IEEE Communications Surveys & Tutorials, Vol. 19, Iss. 1, 2017, pp. 550-586.

[9] F. Domingo-Perez, J. Lazaro-Galilea, A. Wieser, E. Martin-Gorostiza, D. Salido-Monzu, A.d.L. Llana, Sensor placement determination for range-difference positioning using evolutionary multi-objective optimization, Expert Systems with Applications, Vol. 47, 2016, pp. 95-105.

[10] Francisco Domingo-Perez, Jose Luis Lazaro-Galilea, I. Bravo, A. Gardel, D. Rodriguez, Optimization of the Coverage and Accuracy of an Indoor Positioning System with a Variable Number of Sensors, Sensors, Vol. 16, Iss. 6, 2016, pp. 934.

[11] G. Frenkel, Geometric dilution of position (GDOP) in position determination through radio signals, Proceedings of the IEEE, Vol. 61, Iss. 4, 1973, pp. 496-497.

[12] Global Positioning System: Theory and applications. Vols. 1 & 2, Global Positioning System: Theory and applications.Vols.1 & 2, 1996.

[13] E.D. Kaplan, C.J. Hegarty, Understanding GPS : principles and applications, 2nd ed. Artech House, Boston, 2005.

[14] C. Barber, D. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Transactions on Mathematical Software (TOMS), Vol. 22, Iss. 4, 1996, pp. 469-483.

[15] K. Hormann, A. Agathos, The point in polygon problem for arbitrary polygons, Computational Geometry: Theory and Applications, Vol. 20, Iss. 3, 2001, pp. 131-144.

[16] J. Pearl, Heuristics : Intelligent search strategies for computer problem solving, Addison-Wesley, Reading, Mass, 1984.

[17] K.A. De Jong, An Analysis of the Behavior of a Class of Genetic Adaptive Systems, University of MICHIGAN, 1975.

[18] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by Simulated Annealing, Vol. 220, 1983, pp. 671-80.

[19] O. Bozorg-Haddad, M. Solgi, H.A. Loaiciga, Meta-heuristic and evolutionary algorithms for engineering optimization, Wiley, Hoboken, New Jersey, 2017.

[20] T. Bäck, Evolutionary algorithms in theory and practice evolution strategies, evolutionary programming, genetic algorithms, Oxford University Press, New York, 1996.

[21] Punctuated Equilibria: A Parallel Genetic Algorithm, University of Virginia, Department of Computer Science, 1987.