Graph-theoretical Considerations in the Design of Complex Engineering Systems for Robustness and Scalability

by

Gergana Assenova Bounova

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2005

Author . . . .
Department of Aeronautics and Astronautics
January 31, 2005

Certified by . . . .
Olivier L. de Weck
Assistant Professor of Aeronautics and Astronautics and Engineering Systems
Thesis Supervisor

Accepted by . . . .
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Graph-theoretical Considerations in the Design of Complex Engineering Systems for Robustness and Scalability

by

Gergana Assenova Bounova

Submitted to the Department of Aeronautics and Astronautics
on January 31, 2005, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

System optimization for extensibility and robustness is a fundamental challenge of engineering disciplines. Traditional approaches have aimed to optimize cost and performance of a system at a given point in its lifespan. However, as systems evolve with increasing resources and load, system extensibility has to be included in the earliest stages of planning and deployment.

In this thesis, we study the staged deployment of large telescope array configurations as an optimization problem subject to cost, performance and network robustness. The LOFAR (LOw Frequency ARray) is the world's largest telescope array, deploying in its full design 25000 antennas over 350km in diameter in Northern Europe. These are deployed in clusters, and planned to be built in stages, with current funding allowing for 15000 arrays over 100km. This new generation of telescope arrays requires new system design principles and modelling techniques.

We develop a staged optimization framework for modelling network behavior, robustness, and extensibility. We represent large telescope arrays as generalized networks, with nodes as the telescope stations and edges as the cable links between them. We model network design and growth with both graph-theoretical and physical metrics pertaining to imaging properties of each array configuration. Additionally, we model the probability of failure of each topology, both from environmental conditions and random events along the baseline. We make recommendations about the best cost-performance and robustness trade-off configurations.

We introduce two staging principles for system deployment and configuration: (1) a backward approach, in which the design is optimized in the future and scaled down for the initial stages, and (2) a forward approach which achieves optimality at the start and grows the system using an optimization technique. We systematically compare the two staging techniques in the context of telescope arrays and evaluate the hypothesis that the backward approach is superior for telescope arrays because it incorporates knowledge of the desired future end state of the system.

The modelling framework introduced is applicable to various engineering problems suceptible to network representation, including biological systems, telecommunication networks, transportation routes, and space exploration systems.

Thesis Supervisor: Olivier L. de Weck
Title: Assistant Professor of Aeronautics and Astronautics and Engineering Systems

Acknowledgments

My interest in complex systems appeared during a concurrent engineering design project I took part in as an undergraduate at MIT. The sheer size and complexity of systems such as even a single spacecraft made me wonder whether rigorous design theory can be developed to be applied to more than one problem at once. About a year later, I first learned about network theory and started thinking about networks, design, growth and robustness at the same time. Choosing the telescope arrays as an application to my interest I largely owe to the MIT *Multidisciplinary System Design Optimization* course taught by Prof. Olivier L. de Weck and Prof. Karen Willcox. This class also provided me with the necessary set of optimization tools, like genetic algorithms and simulated annealing, but most of all, it developed my *optimization thinking.* Thanks to the various practical exercises, I was later able to write my own algorithms and customize general optimization toolboxes.

The modeling framework in this thesis is entirely based on Babak Cohanim's Master's thesis, which has been a pleasure to build on. I found his work well-organized, his thinking rigorous and his arguments well-defended. His hard work has been an inspiration driving the progression of mine.

Probably the largest impact on my research progress has been cast by Prof. de Weck's guidance. I highly appreciated not only his advice and ideas about this research, but in general his ability to show me the wider context of my project and his shared interest in overarching theoretical principles. I am also grateful for his patience and open-mindedness to my sometimes unconventional ideas, which has greatly helped my creativity.

I am thankful to those people at MIT who supported me and believed in me during my time at the Institute, to my friends and my family who have kept my insanity at a reasonable level, and most of all to my miraculous little brother, whose love, humor and endless energy often charged my batteries during the final stretches of my work.

Contents

List of Figures

List of Tables

Nomenclature

m,n - number of stations for two subsequent stages, $m < n$

(x_i, y_i) - station coordinates

d - site diameter

(u_k, v_k) - uv point coordinates

λ - observation wavelength, [km]

e_{ij} - edge connecting nodes i and j

A - $n \times n$ adjacency matrix for a single array

A_L - $n \times n$ adjacency matrix whose entries set to 1 are replaced with the lengths of the corresponding edges.

X (or R or Γ) - Design Vector (i.e. Design, Architecture, Configuration)

E - System Energy (i.e Objective Function Value)

T - System Temperature

Δ - Difference in System Energy Between Two Design Vectors

w_1, w_2 - weights in the energy function

C - clustering coefficient

L - average minimum path length

Chapter 1

Introduction

Multiple antenna radio astronomy is also known as interferometry. This technique involves using a number of antennas linked together to create an aperture which resolves features with much smaller angles and thus probes deeper into space. Various large telescope projects exist from relatively smaller sizes, like the Very Large Array (VLA) in New Mexico, to Very Long Baseline Interferometry (VLBI) projects, in which stations are too far to be linked with conventional cable, and the data is recorded, transported remotely and then correlated. The applications of such grand-scale astronomy are far-reaching from deep space observation to profound knowledge about the Earth itself. Some scientific results due to the VLBI include the motion of the Earth's tectonic plates, definition of the celestial reference frame, understanding the deep structure of the Earth, creating better atmospheric models, imaging high-energy particles being ejected from black holes and imaging the surfaces of nearby stars at radio wavelengths [1][2].

Figure 1-1: Very Large Array, Socorro, New Mexico, (image courtesy [1])

The design of these large systems with such wide scientific applications is very costly and difficult to plan. In this thesis, we look at the staged deployment of large telescope array configurations as an optimization problem subject to cost, performance and network robustness.

1.1 Motivation and Problem Formulation

LOFAR started as an innovative effort to create a breakthrough in sensitivity for astronomical observations at radio-frequencies below 250 MHz [14]. The radio astronomy technology planned is unchanged since the 1960's: large mechanical dish telescopes collect signals to be sent to a receiver and analyzer. Half of the cost in designing such systems lies in the steel and moving structure. This is why single apertures much larger than the equipment are unaffordable. The answer are large telescope arrays, which can collect signals separately and interpret the image as a whole. In the case of LOFAR, there are numerous simple stations, on the order of 25000 in the full design, organized in clusters, and planned to be built in stages. Phase 1 is currently funded with 15000 antennas on maximum baseline of 100 km.

As envisioned such a complex system has not only to be designed to reach scientific objectives at affordable cost, but also designed to be extensible. Extensibility indicates the capability of the system to increase performance under an increased load when more resources are added. Moreover, an array with a diameter on the order of hundreds of kilometers with thousands of stations has a large probability of failure, either due to equipment resisting environmental conditions or random events along the baseline. To ensure network resilience and sustained functionality even after random failure, the array needs to be designed both for extensibility and robustness.

To meet the above objectives, we modeled large telescope arrays as generalized networks, with nodes as telescope stations and edges the cable links between them. Their design and growth are modeled with both graph-theoretical and physical metrics pertaining to imaging. Recommendations are made about the best cost-performance and robustness trade-off configurations.

1.2 Previous Work on System Scalability and Robustness

1.2.1 Staged deployment and orbital reconfiguration of satellite constellations

Chaize studies the value of flexibility in design in an uncertain environment in the context of the history of the Iridium constellation [4]. The Iridium company believed it could attract about 3 million customers, but was not designed to accommodate the fact that by the time it was deployed the marketplace was already transformed by the growth of terrestrial cellular networks. Thirteen months after beginning operations, Iridium filed for Chapter 11 protection. Chaize argues that this is a consequence of the traditional way of designing large capacity systems which optimizes design for a specific objective (global capacity) and fails to account for demand uncertainty. He proposes a *staged deployment* strategy in which the system capacity is increased gradually and design decisions are made dynamically at each deployment stage. Technically, the value of such built-in flexibility is accessed using real options theory and a multi-objective optimization framework.

Staged deployment is a particular way of introducing flexibility in the system. The goal of Chaize's work is to compare the life cycle costs obtained with flexibility with those obtained using traditional design techniques. He presents the difference between the cost of the traditional design and the expected life cycle cost obtained with staged deployment as an approximation of the maximum price designers should be willing to pay to embed flexibility into the design. In conclusion, Chaize shows an improvement in the life cycle cost on the order of 30% for LEO constellations of communication satellites and suggests that this framework can be used for designing other large complex systems.

1.2.2 Studies on network topology, robustness and cost

There are more references than can be mentioned here about network topology and its implications for cost, performance and robustness, all in various domains. Probably the most numerous are the ones with applications in network biology. Two very extensive reviews in *Nature* [19] and in *SIAM Review* [8] were used for terminology and reference on network theory. Various idea have also spawn from reading studies about the topology and robustness of the Internet [18]. The studies presented below, have been selected because they are more closely related to our research.

Yang and Kornfeld [5] study the optimality of the hub-and-spoke principle for FedEx delivery problem. They use mixed integer programming, modeling the network of next day air cargo delivery flights between a fairly small number of city pairs. It is shown that hub-and-spoke is not always the desired architecture, but the topology varies with the demand, aircraft type and city locations. Only aircraft flight-time operating costs are considered, and no robustness with respect to uncertain demand. The original intent of the study was to model the FedEx delivery network with 26 cities, but it was found that the integer programming algorithms could not handle that many variables. This study is an indicator that understanding network topology is crucial to airlines costs modeling and operations, and also there is a need for heuristic algorithms that can deal faster with large nonlinear design spaces.

A lot of research is devoted to airline flight scheduling, using linear programming flow and search algorithms. Network topology investigations in airline operations are more recent. An example is a study done jointly with the American Airlines Operations Research Department and the Georgia Institute of Technology [6]. The authors review the flights network robustness issue due to canceling flights in operations because of disruptions. Canceling a single flight, means also canceling a sequence of flights that starts and ends at the same airport. It is claimed that fleet assignment and aircraft rotation with short cycles will be less sensitive to cancelations. In the paper, the lower bound for the number of cycles is estimated using the hub connectivity of the fleet assignment. They show that solution models perform better than traditional assignment models.

The last review is of a very similar study to ours with a different goal: to find out how small worlds arise [7] (see Section 1.3). The authors study whether the small-world topology arises as a consequence of a trade-off between maximal connectivity and minimal wiring. To investigate this question, they perform a single-stage simulated annealing optimization with two opposing metrics: connectivity, modeled as the physical (Euclidean) distance between any two nodes and wiring, modeled as the average distance between all pairs of vertices. The optimization goal is to minimize an energy function which is a weighted sum of the two metrics. The optimizations are seeded with k-regular graphs[1]. The authors claim that small-world behavior arises when the two metrics are given equal priority and show results of hub emergence and evolution for varying weights. This claim is proven by evaluating the generated networks with graph statistical measures. The interesting conclusion from this study is that optimization in the tension of two opposing metrics does give rise to scale-free and small-world networks.

1.3 Review of Graph Theory and Network Optimization

Graph theory began in 1736 with Euler's visit to Koenigsberg. People were wondering whether one can traverse all the bridges of Koenigsberg by crossing each one of them exactly once.[2] Ever since Euler and the modeling terminology he introduced, similar and more complicated questions have been raised which perpetrated graph theory development. More recently, a related applied discipline has been evolving quickly, network theory. Network theory, a branch of applied mathematics, shares the goals and knowledge of theoretical graph theory, but studies physical, natural or artificial, networks. Applications include biological networks, like metabolic pathways, gene expression, protein interaction, food webs; transportation networks, like airline routes and destinations and railway tracks; and various large and complex engineering systems, like satellite constellations, electric power grids, cell phone transmission nets, large telescope arrays and so on. Network models apply to many systems that can be observed around us, and can be largely classified in four groups: social networks (social contacts, friendships, scientific collaborations), information networks (Internet, paper citations, semantic networks), technological (power grid, roads network, electrical circuits) and biological networks (foodwebs, protein interaction etc). Different models have been

[1]Graphs in which every node has exactly k neighbors are called k-regular

[2]A walk which starts and ends at the same vertex while passing through every edge only once is called a *Eulerian cycle*. A *Hamiltonian* cycle is a closed path which passes through every node twice. This is also called a *travelling salesman tour.*

developed for each domain, with distinctions and a lot of commonalities. The purpose of complex network theory is to better understand the structure and properties of physical networks and to use this knowledge to make them more resilient to disturbances. For example, social networks research can help well-targeted vaccination to prevent spreading disease and epidemics.

1.3.1 Network terminology

A ***network*** is a collection of nodes, with links between them. ***Nodes*** can be abstract points in interaction space, or geometric points in two, three or more dimensions. An ***edge*** is defined as a pair of vertices from the network. Graphically, it is usually represented by a link between the two vertices. The pair of vertices can be ordered, which corresponds to a directed edge (Figure 1-2 right), or unordered, corresponding to a undirected (or bidirectional) edge (Figure 1-2 left). A graph can be represented with an adjacency matrix. For a n-node graph, the *adjacency matrix* is a $n \times n$ matrix A in which $A(i,j) = 1$ if the i^{th} and j^{th} nodes are adjacent (linked) and $A(i,j) = 0$ otherwise.

One of the striking discoveries in network theory is that there are measures and characteristics

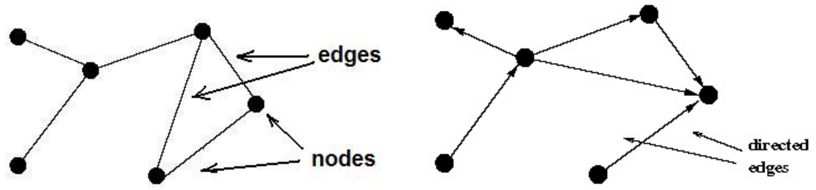


Figure 1-2: Example of a undirected (left) and a directed (right) general graphs; the directed graph is acyclic, the undirected graph contains a loop

that do not depend on the network size, but its structure, therefore they allow comparisons among systems of different size and nature. One of the important characteristics of a graph is the degree distribution. For an undirected graph the *degree* of a vertex is defined as the number of links adjacent to the vertex. For a directed graph, the *in-degree* is the number of edges that originate at the vertex and the *out-degree* the number of edges that terminate at the vertex. A degree distribution is the histogram of vertex degrees. A lot of network properties are related to the degree distribution, including network robustness which is a major concern of this thesis. Figure 1-3 shows three different

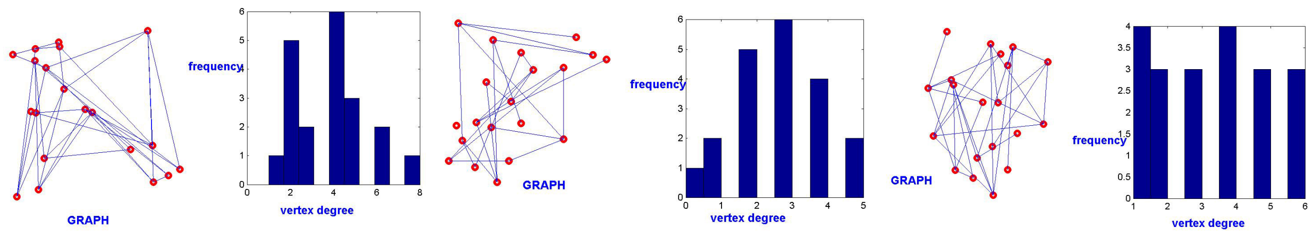


Figure 1-3: Three random graphs with different degree distributions: (a) random with no apparent hierarchy, (b) normal-like, indicating some scaling laws, and hub formation, (c) uniform, which is typical for k-regular graphs.

degree distributions, random, normal-like and uniform-like.

A network is an instance of a connected graph, a mathematical object, in which vertices and nodes have some physical meaning. A *connected* graph is a graph in which every node can be reached from any other node following a path of links (A *path* is an ordered sequence of links, each two consecutive adjacent to each other). It is the rising interconnectivity and interactions in physical systems that perpetrated the study of physical networks in the first place.

Properties and measures about the edges of a graph are mostly related to connectivity and network navigation. Sometimes edges are weighted to show relative importance of the link, for example how well people know each other, how fast the connection is along a certain route or how far the edges are. Edge length, also called cable length is one of the metrics used for evaluating telescope array designs in this thesis. To further explain connectivity some more graph theoretical definitions are needed.

A cycle is a path which starts and ends at the same node (also a closed path). An acyclic graph is graph without cycles. A *tree*" is an acyclic and connected graph. A *spanning tree* of a graph is a subset of $n-1$ edges that form a tree. A *minimum spanning tree* is a set of $n-1$ edges of minimum total weight which form a spanning tree. (When a graph is unweighted, any tree is a spanning tree. A minimum spanning tree, also called *Steiner tree* in this thesis is a spanning tree with a minimum total edge length (sum of all edge lengths or also called total cable length). Many algorithms are used to compute minimum spanning trees or their approximations), since in general this is a NP-complete problem [10]. We use a similar clustering algorithm to the Matlab function *linkage*, in which the closest link to a cluster of points is computed when a new point is considered and so on, iteratively. The minimum spanning tree problem is described further in the following subsection.

1.3.2 Essential network algorithms

Shortest path problem

Shortest paths are applicable to vehicle routing and communications problems and are often useful as embedded in other network navigation algorithms. A famous solution to finding a shortest path between two nodes is given by Dijkstra's algorithm. In Dijkstra, a starting node is chosen and all distances from it to other nodes are set to infinity. As the algorithm *walks* on adjacent edges in a breadth-first fashion, each time it reaches a new node, it relabels the learned distances along the way until it walks the entire graph. Our shortest path algorithm was inspired by [9] (see Section B.1).

This algorithm only works with acyclic graphs. Since we only design minimum spanning trees, no loops (closed paths) exist in the telescope arrays, and hence no two ways to get from station to station. The only path found by the algorithm is the shortest path.

Minimal linkage algorithm

Extending the telescope arrays means adding new stations and linking those stations to the already existing cluster. Optimal linking requires the end array to be a Steiner tree, just like the original one. Our algorithm achieves that by computing the minimum distance to the existing cluster for every new station added B.2.

1.3.3 Network models

Network models have been developed derived from various physical networks of interest with the goal to find similarities between different physical systems and to use the same tools to study their behavior. Three useful models have become popular, from the Erdös-Réniy random graph model, to the Barabási-Albert scale-free model and finally the hybrid network model.

The ***Erdös-Réniy random network*** models was first developed in the 1960s by the two mathematicians Réniy and Erdös. Various improvements and modifications have been done since [8]. The simplest Erdös-Réniy model is to take a n-node graph and connect each two vertices with some probability p. This graph is labeled $G_{n,p}$. In the limit of large n and fixed k it has been found that the probability of any vertex having degree k is $p_k = \left(\begin{array}{c} n \\ k \end{array}\right) p^k(1-p)^{n-k} \simeq \frac{z^k e^{-z}}{k!}$. Various values of p between 0 and 1 create different graph structure, determine connectivity, existence of a large component and so on. The Poisson (because of the Poisson degree probability for large n) random graph model has been studied rigorously and developed to sophistication but not found to be too practical. Most networks, natural or designed, are not random in nature.

The ***scale-free networks*** idea proposed by Barabási-Albert [11] has attracted much attention. These networks are characterized by a power-law degree distribution, that is the probability that a node has degree k is $P(k) \sim k^{-\gamma}$, where γ is the degree exponent. Another way to define scale-free networks is through their evolutionary origin. A popular model is network growth following a preferential attachment law. *Preferential attachment* means that when a new node is introduced, the probability that it is attached to a given node is proportional to the degree of that node. Thus, well-connected nodes keep higher number of links with network growth. The rich get richer is a principle also observed in social networks. People tend to associate with people who have many contacts already. The probability that a node is highly connected plays a larger role than in random graphs which results in a small number of highly connected nodes also known as *hubs*.

Many physical networks have been found to have a power-law, and studies have determined experimentally and with extrapolation the degree exponent. For a small γ hubs are very important (stand out), while for $\gamma > 3$ hubs are irrelevant; for $2 < \gamma < 3$ hub hierarchy forms, and for $\gamma = 2$ a hub-and-spoke formation emerges. Hub-centered, hub-and-spoke and a uniform graph are shown in Figure 1-4. Notice that all three will have different degree distributions, and probably the right-most graph on Figure 1-4 has a distribution which looks like the right-most plot on Figure 1-3.

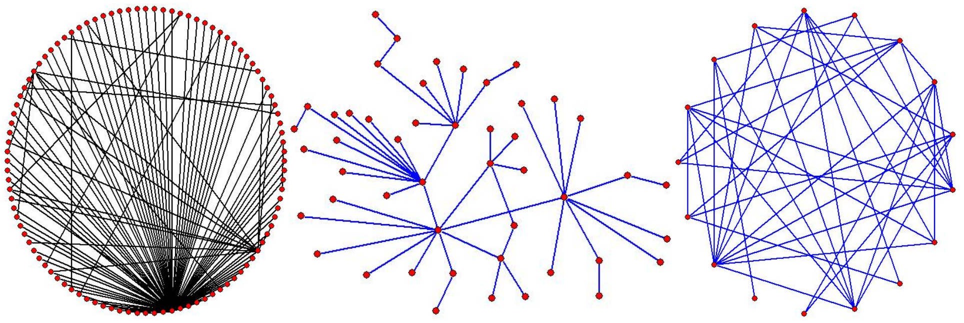


Figure 1-4: Left: hub-centered topology, the majority of vertices have low degrees; middle: hub-and-spoke configurations, characterized by a few well-connected hubs, each forming a hub-centralized component; right: a graph with a uniform degree distribution, all nodes are equal, no hub emergence. These graphs were generated with *Pajek* [24].

Modularity and hierarchical structure add another level of complexity which is not explained by the scale-free model. Local clustering and scale-free topologies are common in real networks and can be generated in an iterative manner from scale-free networks. The starting point is a cluster of densely linked nodes. Next this cluster is replicated few times and the central node is connected to the external nodes of the replicated parts. This preserves the scale-free nature of the network (power-law degree distribution), but exhibits different clustering patterns, different clustering coefficient.

1.3.4 Network growth, network resilience

Network robustness, as reviewed in the literature is presented in detail in Chapter 4, which discusses the robustness of telescope arrays. Robustness is the system's ability to respond to changes in the external conditions while maintaining normal behavior. A lot of studies have been performed on network robustness, largely concentrating on robustness in functionality. The literature defines [19][22] two types of robustness to failure, topological and dynamic robustness. Topological robustness refers to damage on the structure of the network, like node and edge failure, and surviving connected components. Functional (or dynamic) robustness maps the loss of functionality due to partial failure. The metrics can have different implications about the network design, and as established, they determine the network topology.

1.4 Summary of Cohanim's Thesis

The next generation of radio telescope interferometric arrays requires careful design of the array configuration to optimize the performance of the overall system. Cohanim [3][12] has developed a framework, based on a genetic algorithm, for rapid exploration and optimization of the objective space pertaining to multiple objectives. He has evaluated a large space of possible designs for 27, 60, 100, and 160 station arrays. The 27 station optimizations can be compared to the well-known VLA case, and the larger array designs apply to arrays currently under design such as LOFAR, ATA, and the SKA. In the initial implementation of this framework designs are evaluated with respect to two metrics, array imaging performance and the length of cable necessary to connect the stations. Imaging performance is measured by the degree to which the sampling of the *uv* plane is uniform. For the larger arrays he finds that well-known geometric designs perform well and occupy the Pareto front of optimum solutions. For the 27-element case he finds designs, combining features of the well-known designs, that are more optimal as measured by these two metrics. The results obtained by the multiobjective genetic optimization are corroborated by simulated annealing, which also reveals the role of entropy in array optimization. The framework is general, and may be applied to other design goals and issues, such as particular schemes for sampling the *uv* plane, array robustness, and phased deployment of arrays.

Our work is largely based on the physical model and single-stage optimization results of telescope arrays developed by Cohanim.

1.5 Thesis Objectives. Map. Hypothesis

This thesis is looking at two major problems, designing telescope arrays for cost and performance to be extensible and designing telescope arrays for robustness. While valuable design recommendations are made, one of the seeked contributions is to tie graph-theoretical knowledge with optimization of large engineering networks. It is our hope that the framework developed is adaptable to various engineering problems, susceptible to network modeling. Another major goal of this work is to study staged deployment from the point of view of two staging principles: a backward approach, in which the design is optimized in the future and scaled down for the initial stages, and forward, which believes in optimality at the start and grows the system using an optimization technique. Our hypothesis is that the backward approach is superior for telescope arrays because it incorporates knowledge of the desired future end state of the system.

This thesis is organized in four major parts. Chapter 1 presented the problem with background, theoretical and scientific motivation and provided a brief review of relevant network theory. Chapter 2 explains in detail the principles, assumptions and algorithms, in other words the tools used in our research framework. Chapter 3 is dedicated to results about the staged design of telescope arrays from the point of view of cost and performance. Two major staging techniques are compared, while discussing the benefits and drawbacks of staging with recommendations for the best design configurations. Chapter 4 concentrates on network robustness implications for the telescope arrays and tries to explore the tension between Pareto optimality and robustness in the context of the two staging principles. Arrays designed for robustness are presented as an alternative to the Chapter 3 results. Finally, Chapter 5 concludes the thesis with open issues, the remaining cases to be studies and other potential applications of the entire framework.

Chapter 2

Staging Optimization Framework

2.1 *Static* Multiobjective Optimization: Cost Versus Performance

A *system* is a physical or virtual object whose behavior or function is a consequence of interactions between its constituent elements. The *design* of a system is the process of conceiving or planning its form or function with a specific goal in mind. Almost always, systems are designed with multiple goals or for multiple purposes. Finding the best design solution to a single goal is challenging enough and is often posed as an *optimization* problem. Optimizing conflicting objectives for a system, like cost, performance, schedule and risk is called *multiobjective optimization*. For example, optimizing a fighter jet for speed means sacrificing fuel efficiency. Thus a faster airplane might be more expensive to operate. This is a classic example of tension between performance and cost [13].

The cost versus performance trade-off in networks is usually closely coupled with the network topology. The telescope arrays design problem deals with positioning both the nodes and the links in the network. The placement of the antennas (nodes) determines the fidelity of the obtained image, but it also affects the cost of building the infrastructure, like the power distribution, site preparation and laying connecting cable. In network terminology, the configuration of the nodes affects the coverage and the connectivity of the network. There are various metrics representing coverage and linkage, and in general all of those are opposing metrics. Intuitively, if a net is very spread out and has good coverage, it has a large diameter, long shortest paths from node to node and overall large total linkage distance. Cohanim [12] modeled coverage as the imaging performance of the array by estimating the number of unfilled *uv* points in the *uv* plane. Cost is represented by the total cable length of the array. The framework developed in [12], based on a genetic algorithm, rapidly explores of the objective space pertaining to the two cited objectives. A solution to the problem of finding an optimal configuration for a fixed number of stations is proposed. Figure 2-1 shows an example results of telescope configurations optimized for minimum cable length (left-most), maximum coverage (middle) and both, equally weighted (right-most). The top plots show the geometry, the bottom ones show the corresponding *uv* density points. The trade-off between cost and performance is clear: low-cost designs have the worst relative coverage, while the best performance designs have the greatest cable length. We call this problem type *static* multiobjective optimization. The rest of this chapter deals with multi-staging, which stretches the baseline questions in time and seeks to find an optimal configuration, which is also designed to grow optimally, subject to the same two objectives, cost and performance. The term we use for this problem type is *dynamic* multiobjective optimization.

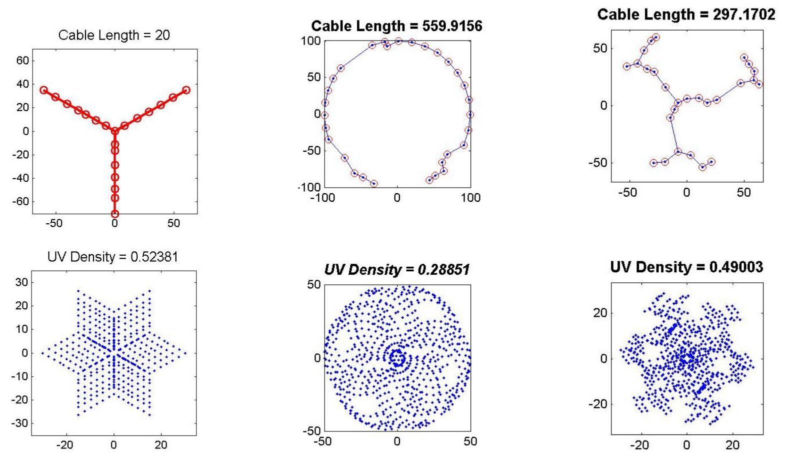


Figure 2-1: Telescope array configurations with 27 stations optimized for : (a) minimum cable length, a VLA-like configuration, (b) maximum coverage or minimum uv density, a circular configuration, (c) both metrics, a randomized log spiral geometry

2.2 *Dynamic* Single-step Staging: Forward Versus Backward Strategy

Complex systems can either scale with time (extend or shrink) or evolve or both. Extensibility (scaling) means preserving the nature of the elements of the system and their function, but increasing their number or size. Evolution means changing fundamental form and/or function. In this thesis, we consider the former scaling a telescope array means adding stations and connecting them to the existing infrastructure without changing it. *Single-step* staging will be defined as designing a m-station configuration and a n-station configuration such that $m < n$ and the n-array contains the m-array. Notice that for this application, the second stage is strictly larger than the first ($n-m \gg 1$) since we are interested in extensibility, not accretion (slow small-scale growth).

Two diametrically opposite approaches were used to design telescope arrays for extensibility: optimizing upon best knowledge of the present, as a forward-looking technique; and deducing what is needed now based upon best estimate of the future, as a backward-looking technique. So, in the forward approach, an array is optimized using a genetic algorithm, developed in [12]. Then, to scale the array, stations are added using a custom-written simulated annealing algorithm. The best second-stage configuration is identified based on some weighted metric of cost and performance. In the backward approach, first an array is optimized using a genetic algorithm, with the desired number of stations for a second-stage. Then, the same simulated annealing algorithm is used to find an optimal subset of the array, to be used as a first stage. Figure 2-2 shows an example of a GA-optimized configuration with its optimal subset chosen for the first stage. The plots on the right shows the first and the second stage with their corresponding uv densities.

2.2.1 Problem formulation

Problem setup: Let $(x_i, y_i), i = 1 \dots m$ and $(x_j, y_j), j = 1 \dots n$, such that $0 \leq \sqrt{x_j^2 + y_j^2} \leq d$, be sets of points in the plane, with $(x_i, y_i) \subset (x_j, y_j)$, and d is a constant diameter. Notice that, $(x_i, y_i) \subset (x_j, y_j)$ means that for all i, there exists a j, such that $(x_i, y_i) = (x_j, y_j)$. Suppose that $e_{i_1 i_2}$ and $e_{j_1 j_2}$ are sets of edges connecting the sets of points (x_i, y_i) and (x_j, y_j). We require $e_{i_1 i_2} \subset e_{j_1 j_2}$, so that the original infrastructure like laid cable, roads, trenches are kept and expenses for recabling and rebuilding the network are avoided. Then, *total cable length* is computed as the sum of the lengths of all edges in the network ($\sum |e_{i_1 i_2}|$ and $\sum |e_{i_1 i_2}|$ respectively). The number of uv points is computed directly from the stations coordinates as $u_k = \frac{x_i - x_j}{\lambda}, v_k = \frac{y_i - y_j}{\lambda}$, for all i, j, where λ

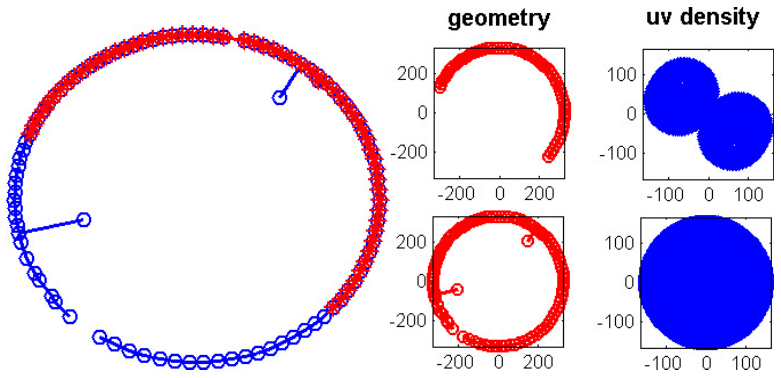


Figure 2-2: Example of backward staging. The filled circles segment in the left plot is the selected optimal subset of the wider GA-optimized array; the right plot shows the first-stage array and the second-stage array with their *uv* density plots

is the observation wavelength. These *uv* points are correlated with the points of a pre-computed uniform grid. The number unfilled points on the uniform grid is presented as percentage to form the *uv* density metric. An example of a uniform grid and the filled *uv* points by a log spiral geometry is given in Figure 2-3.

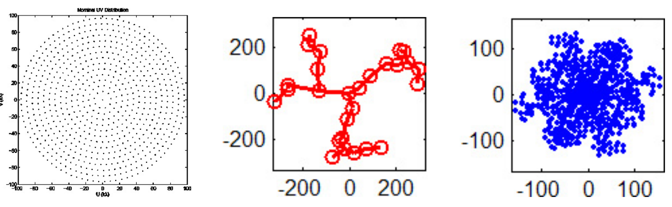


Figure 2-3: A nominal grid (left) used to calculate the filled *uv* points for a log spiral geometry

Problem statement: For given m, n and d, find a set of n points $(x_j, y_j), j = 1 \dots n, 0 \leq \sqrt{x_j^2 + y_j^2} \leq d$ and a subset of m points $(x_i, y_i), i = 1 \dots m, 0 \leq x_i, y_i \leq d$, such that their cable lengths and *uv* densities are minimized, for all such sets and their subsets. A formal mathematical formulation of the problem statement is given by (2.1).

$$\begin{array}{rc}
\textit{find} & \{(x_i,y_i)\}_{i=1\dots m},\ \textit{and}\ \{(x_j,y_j)\}_{j=1\dots n},\ 0 \leq \sqrt{x_i^2+y_i^2} \leq d,\ 0 \leq \sqrt{x_j^2+y_j^2} \leq d \\
\textit{such that} & \{(x_i,y_i)\} \subset \{(x_j,y_j)\},\ m < n,\ \textit{is a connected subset} \\
\textit{and such that} & f(\{(x_i,y_i)\}), f(\{(x_j,y_j)\}), g(\{(x_i,y_i)\})\ \textit{and}\ g(\{(x_j,y_j)\})\ \textit{are minimized} \\
\textit{where} & \mathbf{f}\ \textit{is the array total cable length and}\ \mathbf{g}\ \textit{is the array uv density}
\end{array} \tag{2.1}$$

This problem formulation can be extended to any number of stages, defined as subsets in (2.1). The metrics can also be modified to reflect robustness, flexibility, or different cost and performance models. Priorities (weights) can be assigned to different stages to account for the varying importance of objectives in the present and in the future. Such weighting can have impact not only on the resulting designs, but also on the optimization tools preferred to solve the problem.

2.2.2 Modeling, assumptions

The vast scope of this problem is a mixture of complex scientific objectives (probing deeper into space), using interferometry and electromagnetic radiation to interpret signals, point-design engineering of radio telescope stations and system engineering to understand their operation in a complex network. To optimize such a large system with many interactions and complex objectives, we had to make simplifying modeling assumptions. High-level modeling assumptions are listed below, algorithm specific assumptions are described in the following sections.

- Telescope stations are modeled as points in two dimensions, not apertures, a minimum distance of 1 kilometer apart. In reality, stations can be from 25-meter in diameter radio telescopes stretched out along 21-kilometer arms like the Very Large Array in New Mexico, to many clusters of a total of 25000 telescopes spread out in an area as large as 350 kilometers in diameter, as in the full LOFAR design. Our assumption is valid only if a telescope aperture size is much smaller compared to the site diameter.

- The design space is circular and two-dimensional with a varying diameter from 100 to about 400 kilometers scaled with the array size. Varying the site diameter is implemented in order to enable the comparison between the performance of arrays of different sizes. For configurations with the same number of stations, the same site diameter is used, include for damaged arrays to calculate percentage of lost coverage, the *uv* density metric is compared with the site diameter of the original array.

- The number of stations is fixed at every stage; that is, no optimization is done on the number of stations

- When the array is scaled, (stations are added or removed), the initial infrastructure is kept constant; station coordinates do not change, nor does the linkage between them. This means that as new links are created or links are removed, the resulting network graph is still connected and contains no loops or double edges. *Recabling*, that is changing the topology of the network for the purposes of optimality has been considered, but proved to be too costly and was rejected as an option.

2.2.3 Variables, design vector, constants

There are problem-specific and algorithm-specific variables and constants. The most important problem-level constants are the number of stages and the number of stations per stage, decided upfront. Two to three stages have been simulated with small and large number of stations. For benchmarking purposes, 27 stations are chosen for the first stage (matching the VLA design), 60 for second stage and 99 for third, which follows the design plans for the LOFAR [12],[14]. For testing algorithms and convergence limits lower numbers of stations are used: 6, 15 and 24 for first, second and third stage respectively.

The design space is a circle with a constant diameter. The diameters for different numbers of stations are computed proportionate to 400 kilometers corresponding to 60 stations. This allows more design space for larger arrays. This reference 400-kilometer site diameter approximates LOFAR planning specifications [14]. Other problem-specific constants are the observation wavelength λ, set to 4 meters, the choice of a radial *uv* distribution and the initial geometry seeds, inputs to the genetic algorithm. The geometric seeds can be VLA-like (Y-shaped), circular, triangular, Reuleaux triangular, log-spiral and purely random. The rest, algorithm specific variables and constants are described in the following sections.

The design vector consists of telescope stations coordinates $(x_i, y_i), i = 1 \ldots n$, where n is the number of stations. Therefore, if the array has n telescopes, the design vector contains $2n$ variables.

2.2.4 Metrics, design objectives

Two design objectives for the telescope arrays are mapped to the traditional cost and performance metrics: cable length and *uv* density.

Cable length

Building the array involves expenses for building and positioning the telescope stations and laying the expensive digital optical fibre. We model the cost involved with the total length of the cable

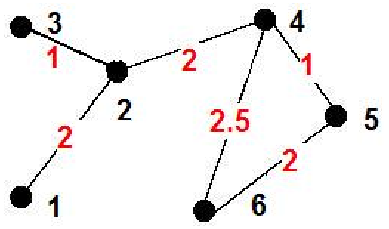


Figure 2-4: Example of a undirected cyclic graph with arbitrarily assigned edge weights

needed. For array diameters on the order of hundreds of kilometers, the added cost is significant (and scales approximately linearly with the size of the array).

In a network, the total cable length is the sum of the length of all the links of the network (graph edges). A convenient way to calculate that is to keep an equivalent of an adjacency matrix, which instead of 0s and 1s contains 0s and lengths ($|r_i - r_j|$) for each $(r_i, r_j) = 1$ entry of the adjacency matrix (equal to its own transpose). We call this the adjacency-lengths matrix. Apparently, it is also a symmetric matrix. Therefore, all edge lengths are recorded twice and the total network length will be the sum of all entries in the adjacency-lengths matrix, divided by 2. For example, consider the graph on Figure 2-4. The adjacency matrix for this graph is 6×6 and is equal to $A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$.

The corresponding adjacency-lengths matrix is $A_L = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 2.5 \\ 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 2.5 & 2 & 0 \end{bmatrix}$. Then the total edge length of the graph in Figure 2-4 is the sum of the entries of A_L divided by 2, which is 10.5.

UV density metric

Modeling the imaging quality of the array can be done in different ways. The *UV Density Method* developed by Cohanim [12], calculates the distribution of *uv* points for a large number of stations. It starts by calculating a nominal grid, which consists of *uv* points placed according to the desired radial distribution, and then spaced apart equally in azimuth at each radius. The basic idea of acquiring a distribution in the *uv* plane is to have all *uv* points roughly the same distance from their neighboring uv points. This is equivalent to having all *uv* points occupy an equal area in the *uv* plane. The actual *uv* distributions are calculated using equations (2.2).

$$u_{i,j} = \frac{x_i - x_j}{\lambda} \qquad (2.2)$$
$$v_{i,j} = \frac{y_i - y_j}{\lambda}$$

An actual *uv* point is associated with the nearest *uv* point in the nominal grid, using a nearest neighbor algorithm from Matlab. If an actual *uv* point is associated with a nominal grid point, then the nominal grid point is considered *filled.* The *uv* metric is expressed as the percentage of unfilled nominal *uv* points, $M = \frac{N_{uv} - N_{uvactual}}{N_{uv}}$.

The downside of using the *uv* density metric to compare arrays with different sizes is that it is normalized to the number of stations ($\frac{n(n-1)}{2}$), so it does not show absolute increase in array performance. One way to circumvent this problem is to use the largest nominal grid (computed for

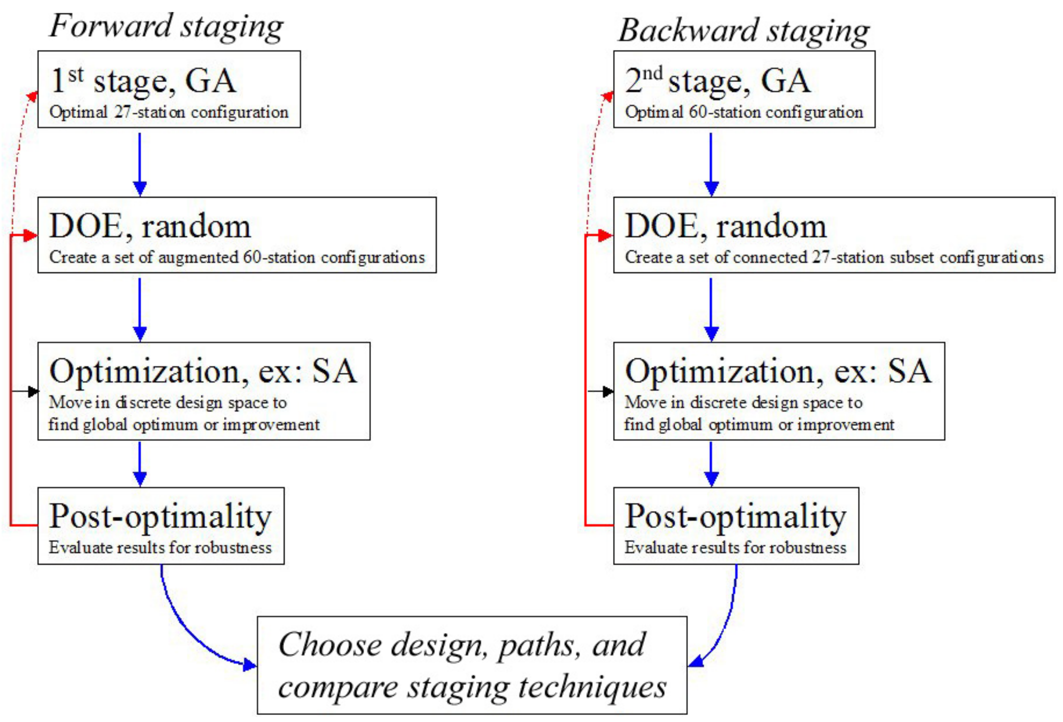

Forward staging
1st stage, GA
Optimal 27-station configuration
DOE, random
Create a set of augmented 60-station configurations
Optimization, ex: SA
Move in discrete design space to find global optimum or improvement
Post-optimality
Evaluate results for robustness
Backward staging
2nd stage, GA
Optimal 60-station configuration
DOE, random
Create a set of connected 27-station subset configurations
Optimization, ex: SA
Move in discrete design space to find global optimum or improvement
Post-optimality
Evaluate results for robustness
Choose design, paths, and compare staging techniques

Figure 2-5: Algorithms sequence flowchart; *forward* and *backward* staging techniques are shown side by side; both techniques use the same sequence of optimization modules, but with different inputs and outputs

third stage), for all stages compared.

Other, more sophisticated, but also more computationally intensive metrics have been proposed. A discussion and list of references are provided in [12]. The two metrics described above were chosen for their simplicity and also for their orthogonality. Minimizing the cable length tends to cluster the array which has an opposing effect to maximizing the coverage. Minimizing the percentage of unfilled nominal grid points means spreading out the telescope stations which increases the total cable length.

2.3 Algorithms, Optimization Tools

2.3.1 Algorithms flowchart

The backward and forward staging principles are executed with the same algorithmic skeleton, but with different inputs and outputs, as indicated on Figure 2-5. Both start with a genetic algorithm routine, then create and sample the design space of the resulting array and finally explore this sample space using simulated annealing. The two- or multiple-stage output (set of arrays) is analyzed for optimality, metric performance, robustness, geometry convergence and other graph statistical properties. After this post-optimization evaluation, the backward and forward approaches are compared per criterium and overall and design recommendations are made.

2.3.2 Algorithms detailed description

The choice of algorithms is essential in obtaining a good solution or gaining insight into the nature of a problem. For problems with highly nonlinear design spaces and objective functions, like this one, it is difficult to visualize the entire design domain, the location and topology of design families, or their evolution at every iteration. With $2n$ design variables and the two highly nonlinear

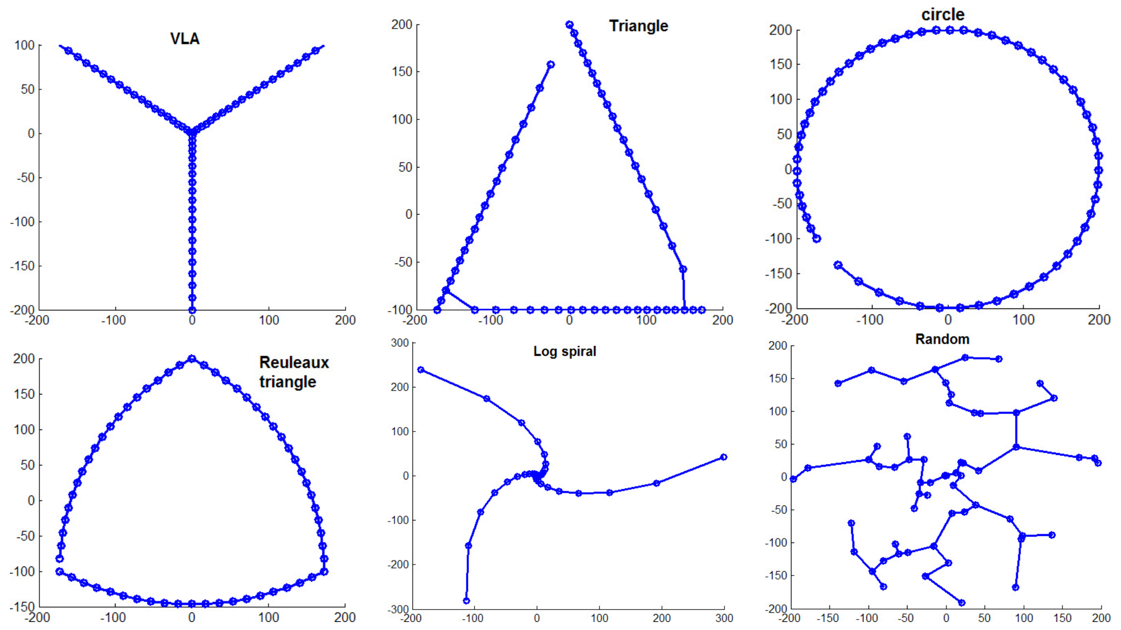


Figure 2-6: Array seed geometries, 60 stations: (top) VLA-like, triangular, circular, (bottom) Realeaux triangular, log spiral and random

objective functions cable length and *uv* density, it is impossible to claim the existence of an optimal solution, except for very small dimensions or inconceivably large number of iterations. Gradient or deterministic algorithms are weak for nonlinear problems, because they get trapped in local minima. Heuristic techniques, such as simulated annealing, neural networks, and genetic algorithms are better at handling nonlinearities, though all of them require greater computational resources. Depending on the physics or the model the design space can be discrete or continuous (the design variables can have continuous or discrete domains). Generally, gradient-based algorithms deal better with continuous spaces, while discrete spaces favor heuristics [13].

The computational framework developed for this thesis is based on a genetic algorithm developed by Cohanim [12] and a custom-written simulated annealing algorithm. Various network navigation algorithms were also developed to support the design space search in the optimization process.

The computational framework is initialized with choices of two key global variables: the number of stages, and the number of stations per stage.

Initialization [12]

The initialization routine creates an initial population of array designs for the first generation of the genetic algorithm. Stations coordinates are assigned according to a pre-selected geometry, which is constrained by the site diameter. Cohanim tested several geometric seeds, like circles, triangles, Reuleaux triangles, VLA-like (Y-shaped) and randomly placed stations, as shown on Figure 2-6. Different geometries fulfill better different objectives and hence can be found in different places on the Pareto front. For example, random (non-regular or non-geometric) array seeds are never Pareto-optimal according to Cohanim, but can converge to regular designs. The initial seeds are essential to the convergence behavior and Pareto-optimality of the entire population. A question we investigate is how the geometric patterns evolve when the arrays are extended and whether the staging principles used induce new geometries or give more insight about optimality. For example, as discussed in Chapter 3, *log spirals* emerged as optimal patterns from the forward strategy, which is why they were added to the seed geometries a posteriori.

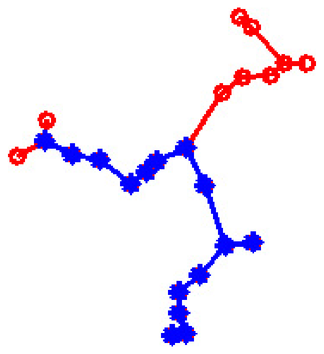


Figure 2-7: A connected 15-subset of a 24-array; the initial arrays is plotted with stars, the additional stations with circles

GA population

A genetic algorithm with tournament selection is used for the optimization of the first stage of the array. The detailed algorithm description can be found in [12]. The first stage is optimized with 200 generations and populations of 100 each for large number of stations ($27 \rightarrow 60 \rightarrow 99$) and 200 designs each for smaller number of stations ($6 \rightarrow 15 \rightarrow 24$). Other constants used in the GA are the crossover rate (0.9), the mutation rate (0.01) and the elitism rate (0.01). The output is an evolved population of designs, with their station coordinates, evaluated objective metrics, non-dominated subset and convergence record. This first-stage optimization step is the same for both the forward and backward techniques. The only inputs that are different are the number of stations for the stage and the design space diameter.

DOE (*Design of Experiments*)

This procedure samples parts of the design space to be searched, evaluated or evolved. For highly dimensional and nonlinear problems, selecting representative parts of the design space can be a challenging task, with its success subject mostly to computational resource constraints. Two fundamentally different DOE principles are presented in this work, associated with the forward and backward techniques.

Backward staging: The GA output in this case is a population of second-stage n-arrays. The purpose of the DOE is to find, for each array, a connected m-subset (or a set of m-subsets) which is a good starting point for the simulated annealing algorithm. Figure 2-7 gives an example of a connected 15-subset of a 24-array. Ideally, the number of all subsets is $\begin{pmatrix} 24 \\ 15 \end{pmatrix} = 1307504$, however, the actual size of the search space is much smaller because not all of these subsets are connected. The purpose of next stage optimization is to find the best such subset subject to minimizing some weighted combination of uv density and cable length.
The essence of this DOE algorithm is to *walk* along the original array, that is for a given original node, to look for neighboring nodes (connected via an edge) and so on to create trees of the desired subset size. The detailed algorithm is attached in B.3.

Forward staging: The GA-output in this case is a population of first-stage m-arrays. The purpose of the DOE is to search a sample of randomly-generated stations to find a $(n-m)$-subset such that added to the original m-array, creates a n-superset, which is a good starting point for the next stage optimization. Figure 2-8 gives two examples of connected 15-supersets of 6-arrays. The DOE algorithm searches randomly a large sample of stations and selects the best found set of $n-m$ stations to add to the original m-array to create a n-array. In this case, the size of the design space is always $\begin{pmatrix} n \\ (n-m) \end{pmatrix} = \begin{pmatrix} n \\ m \end{pmatrix}$. The entire algorithm is presented in B.4.

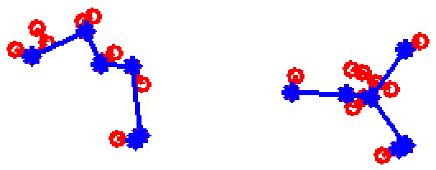


Figure 2-8: Two 15-supersets of 6-arrays; the initial arrays are plotted with stars, the additional stations with circles

SA

Simulated annealing is a heuristic method (using rule of thumb rules or common-sense observations and probabilistic moves to look for good solutions) which is a mathematical analogy of the cooling of set of atoms to a state of minimum energy. The basics of the SA have not changed much since the method was introduced [15]. According to statistical mechanics, if a liquid cools and anneals too quickly, then the material solidifies in a sub-optimal configuration. If the process is slower, the liquid will solidify optimally into a state of minimum energy. This ground state corresponds to the minimum of an objective function in optimization. In our framework, this function is called *energy function* and is a weighted sum of the two objective metrics, cost and performance, $E = \lambda(UV_{dens}) + (1-\lambda)(L_{cable})$.

A general simulated annealing procedure is outlined below [16]:

Terminology:
X - design vector, **E** - system energy (i.e. objective function value), **T** - system temperature, Δ - difference in system energy between two design vectors

Simulated annealing algorithm:

1. Choose a random X_i, select the initial system temperature, and specify the cooling (i.e. annealing) schedule
2. Evaluate $E(X_i)$ using a simulation model
3. Perturb X_i to obtain a neighboring Design Vector (X_{i+1})
4. Evaluate $E(X_{i+1})$ using a simulation model
5. If $E(X_{i+1}) < E(X_i)$, X_{i+1} is the new current solution
6. If $E(X_{i+1}) > E(X_i)$, then accept X_{i+1} as the new current solution with a probability $e^{-\Delta/T}$ where $\Delta = E(X_{i+1}) - E(X_i)$.
7. Reduce the system temperature according to the cooling schedule.
8. Terminate the algorithm or iterate 3-7 if termination criteria not met.

The algorithm parameters varied to accommodate the convergence properties of the problem. An exponential cooling schedule was used at all times with different cooling steps. The most common scenario employed has a starting temperature of 1000, and cooling step of 1.01 ($T_{i+1} = T_i/1.01$). The algorithm is terminated either when the energy becomes lower than 10% of the starting energy or when the temperature reaches a minimum. A lower bound of 10^{-6} was used in most cases, though convergence plots showed that a minimum of 10^{-3} was often sufficient. Notice that 10% of the starting energy is practically unachievable, so this condition acts as an *ad infinitum*, making the termination criterion entirely a function of the number of iterations.

Two instances of this algorithm have been developed for backward and forward staging with different energy evaluation and perturbation steps. While energy evaluation is almost the same for the backward and forward principles, the design vector perturbation is a completely distinct procedure. In Table 2.1 the two are described side-by-side.

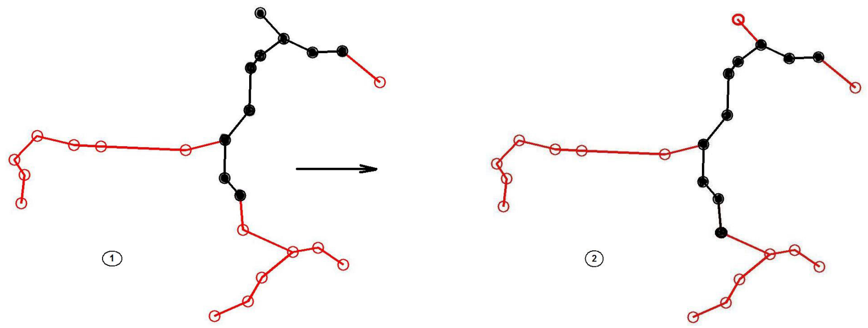


Figure 2-9: Perturbing the design vector from configuration (1) to (2); the goal is to find an optimal 10-station subset of the 25-station array; the second configuration is formed by removing a station and adding a new one to the original connected subset; this example illustrates that there is only a limited number of connected subsets of any connected graph

Energy function
Consider a case with m stations for the first stage and n for the second ($m < n$). The first-stage m stations are always a subset of the second-stage n, regardless of the construction process. Let $P(\bar{x}, \bar{y})$ be the performance (uv density) metric for a given array and $C(\bar{x}, \bar{y})$ be the cost (total cable length) metric. The energy function is computed as $E = w_1 P + w_2 C$, where $0 \leq w_1, w_2 \leq 1,\ w_1 + w_2 = 1$ are priority weights. To avoid scaling problems, both metrics are normalized by the mean of a randomly generated population prior to the optimization. The energy function then always assumes a value around 1.

Perturbing the design vector
Perturbing the design vector requires special attention since it is different for the backward and the forward path cases. For the first, a subset of an array is perturbed to another (adjacent) subset, but removing one station and adding another, while making sure that this procedure produces a connected subset. This means that every time a station if checked for removal, the algorithm verifies that this removal will not produce a disconnected graph. A subset of this algorithm is the shortest path graph algorithm discussed in Chapter 1. An example of a perturbation step is given in Figure 2-9.

In the forward path case adding new stations, involves finding new optimum cabling while keeping the old links. The general linking problem is discussed in detail in [12], where a solution to the Steiner problem is implemented to find the optimal linkage for a geometrically fixed network. In order to add new edges, we wrote a new linkage algorithm using the same principle of clustering linked stations and connecting the new point to the closest point of the cluster. With every new edge the adjacency-lengths matrix is updated, so that the new cable length can be computed.

The metrics, uv density and cable length, are calculated from the network information as explained previously, at each step of the simulated annealing routine. The energy function is a weighted sum of the two, where the weights attach relative importance to the metrics. Experiments with different weights were performed to explore different geometries and to construct a full Pareto front. Most of the simulations and analysis are performed with equal weights ($w_1 = w_2 = 0.5$).

Post processing

Post processing and results analysis required the development of additional algorithms to calculate statistical properties of the graphs like degree distributions, shortest and average path lengths, node centrality metrics, clustering coefficients. Further programs were written to model node and edge attacks to estimate robustness. Some of former metrics will be introduced formally alongside with the analysis, while the robustness models are explain in detail in Chapter 4.

Visualization is another tool which has greatly contributed in understanding the results. Numerous programs have been written to extract the essential from MatLab data file and plot Pareto

Table 2.1: SA algorithm steps for backward and forward staging techniques

Backward SA	**Forward SA**
1. Start with a GA-optimized n-array and a m-subset; set initial temperature and cooling step	1. Start with a GA-optimized m-array and a n-superset; set initial temperature and cooling step
2. Evaluate the energy $E(\bar{x}_i^m, \bar{y}_i^m)$ of the m-array	2. Evaluate the energy $E_i(\bar{x}_i^n, \bar{y}_i^n)$ of the n-array
3. Perturb the m-array by removing and adding a station from the n-array, while keeping the $m - array$ connected	3. Perturb the n-array by choosing another set of $n-m$ stations complimentary to the base m-array from a randomly-generated larger sample
4. Evaluate the energy $E(\bar{x}_{i+1}^m, \bar{y}_{i+1}^m)$ of the new m-array	4. Evaluate the energy $E_i(\bar{x}_{i+1}^n, \bar{y}_{i+1}^n)$ of the new n-array
5. If $E(\bar{x}_{i+1}^m, \bar{y}_{i+1}^m) < E(\bar{x}_i^m, \bar{y}_i^m)$ then keep $\{\bar{x}_{i+1}^m, \bar{y}_{i+1}^m\}$ as the new solution	5. If $E(\bar{x}_{i+1}^n, \bar{y}_{i+1}^n) < E(\bar{x}_i^n, \bar{y}_i^n)$ then keep $\{\bar{x}_{i+1}^n, \bar{y}_{i+1}^n\}$ as the new solution
6. If $E(\bar{x}_{i+1}^m, \bar{y}_{i+1}^m) > E(\bar{x}_i^m, \bar{y}_i^m)$ then keep $\{\bar{x}_{i+1}^m, \bar{y}_{i+1}^m\}$ with probability $e^{-\Delta/T}$, where $\Delta = E(\bar{x}_{i+1}^m, \bar{y}_{i+1}^m) - E(\bar{x}_i^m, \bar{y}_i^m)$	6. If $E(\bar{x}_{i+1}^n, \bar{y}_{i+1}^n) > E(\bar{x}_i^n, \bar{y}_i^n)$ then keep $\{\bar{x}_{i+1}^n, \bar{y}_{i+1}^n\}$ with probability $e^{-\Delta/T}$, where $\Delta = E(\bar{x}_{i+1}^n, \bar{y}_{i+1}^n) - E(\bar{x}_i^n, \bar{y}_i^n)$
7. Update temperature according to cooling schedule (*cool_step* = 1.01): $T_{new} = T/1.01$,$T \equiv T_{new}$	7. Update temperature according to cooling schedule (*cool_step* = 1.01): $T_{new} = T/1.01$,$T \equiv T_{new}$
8. If $E(\bar{x}_{current}^m, \bar{y}_{current}^m) \leq 0.1E(\bar{x}_1^m, \bar{y}_1^m)$ or $T < 10^{-6}$ then terminate algorithm and save current solution. Otherwise, iterate.	8. If $E(\bar{x}_{current}^n, \bar{y}_{current}^n) \leq 0.1E(\bar{x}_1^n, \bar{y}_1^n)$ or $T < 10^{-6}$ then terminate algorithm and save current solution. Otherwise, iterate.

fronts, color-code geometries, show configurations, their linkage and corresponding *uv* density. The importance of visualization strategies is discussed in the conclusion chapter.

Chapter 3

Application to Radio Telescope Arrays

The new generation of telescope arrays requires system design skills and understanding of network behavior and properties. The network configuration drives the cost and performance of the system through time, together with its response to failure or changes. Designing for flexibility, robustness and extensibility takes the multi-objective optimization problem a couple of steps further. In this chapter, we present telescope array configurations results from designing for extensibility using a genetic algorithm and simulated annealing.

3.1 Variables Assignment

3.1.1 Design variables

The design vector consists of stations coordinates placed strategically in a circular space with diameter which is proportionate to the size of the array. For 60 stations, the benchmark is 400 km which comes from LOFAR specifications. This means that for 30 stations, for example, the site diameter will be 200 km and so on. Two sets of experiments were performed, with backward and with forward staging techniques. Telescope arrays are optimized in three stages, a 27-station first stage, extended to 60 stations for the second stage and to 99 for the third. The optimization framework is generalized to accommodate any number of stages, but for array sizes as large as 100, simulations with reasonable fidelity take on the order of weeks to converge on a single processor. This is why algorithms testing was performed with smaller number of stations per stage, namely 6, 15 and 24 stations for the three consecutive stages.

3.1.2 Design metrics / objectives

The traditional design objectives, cost and performance, are implemented as total array cable length and imagining quality (*uv* density metric), as described in Chapter 2. These two objectives are conflicting (orthogonal in objective space), since one tends to cluster the stations together (cost) and the other to spread them out (performance). There are various alternative metrics that exhibit the same tension. Some of them can be derived from the physics of the telescope array, others from general graph properties. Usually metrics that couple the physics with the theory are the best candidates to give insights about the design of the system. Therefore, we claim that the two metrics chosen cover both domains and are simple enough to compute and analyze.

3.2 Single-stage Insights by Cohanim [12]

Cohanim performed optimizations for arrays of 27, 60, 100, and 160 stations with various geometric seeds, among which VLA-like, circular, triangular, Reuleaux triangular, hybrid and random. He used a multiobjective genetic algorithm as a tool to search the configuration space for good trade-off solutions. The solutions display a wide Pareto front showing the trade-off between decreasing cable length, and thus decreasing cost, and improving array performance. Along the Pareto front designs range from VLA-like structures to ringlike structures. Nongeometric arrays whose stations are placed randomly are clearly not Pareto-optimal. For the purposes of verifying convergence of the Pareto front, both geometric and nongeometric configurations are seeded into the genetic algorithm. In some cases, the random geometries converge toward regular geometries, but in most cases they stay *behind* the Pareto front (entropy prevails).

Some of the major conclusions from the single-stage experiments concern the fitness of various geometries. It was found that nadir-utopia solutions can be regular geometries, like Reuleaux triangles, or they can be derived from regular geometries, like slightly randomized VLA-like for minimum cable and circular shapes with inward protruding arms for minimum *uv* density, or they can be hybrid of different seeds. An interesting result is that as the number of stations increases, the population evolves away from the Pareto front in objective space. One of the reasons could be that on fixed-diameter site, the cost of adding more stations does not provide the benefit of better coverage due to a saturation effect. We remove the saturation problem by scaling the site diameter with the number of stations. As the design space increases, the nominal grid increases too, therefore the percentage of unfilled *uv* points is calculated on a geographically expanded space.

Several further work suggestions have been addressed in this thesis. Not modeled by Cohanim, phased deployment and on-site reconfiguration have already been planned for projects like the LOFAR and may be useful in finding timely answers to scientific questions or in reacting to the data collected by a first-stage array. Another new aspect is array robustness, which is essential for the array performance over time. Modeling multiple hubs and redundant links is a way to ensure robustness at a greater cost, but maybe for greater benefit. The framework established in this thesis attempts to address these ideas and set the stage for applying its algorithms to other network problems.

3.3 Two-stage Results, $27 \to 60$ Stations

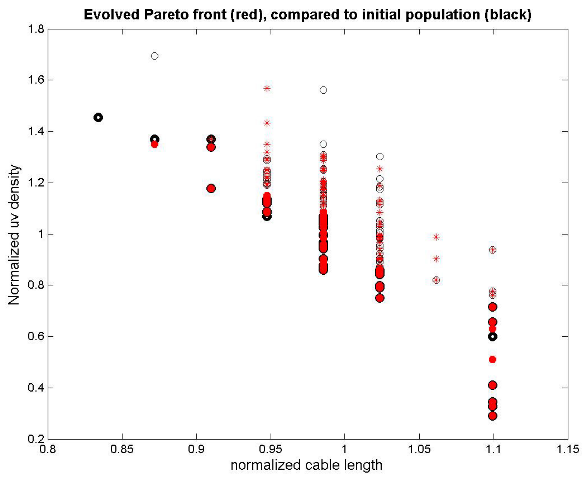
The two stages on which our optimization scheme is based are a GA step, as modeled previously, and a built-on stage, optimized with a SA algorithm. Simulation results are available for both backward and forward staging techniques. In all cases, the SA optimization step is repeated to the second-stage array to optimize a third stage.

3.3.1 Pareto front advancement

Plotting the initial versus the evolved population in objective space shows the Pareto front advancement. The amount and direction of movement of the Pareto front show the relative effectiveness of the algorithm. This analysis also shows how constraints reduce the design space in the backward strategy versus the forward strategy, because improvement depends on the size of the design space. The more options available, the more likely the existence of lower energy states to be explored by the algorithm.

The effect of the connectivity constraint in the backward strategy is expressed as a discreteness in the cable length metric, evident for most small-number-of-stations cases. Figure 3-1 shows discrete slots for all arrays for a case of 6 stations (left) and 27 stations (right). The less the number of stations, the less the number of connected subsets of the original array, hence the less the optimizer can do. Clearly, doing backward SA steps repeatedly pays off only for large telescope arrays, where the first-stage answer is not a priority.

The forward strategy does not have this drawback, since new stations are constantly relinked optimally. This is affordable at the optimization phase because these new links do not exist yet.



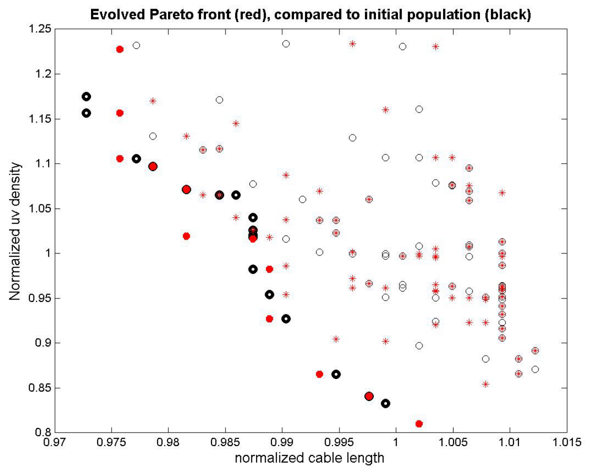


Figure 3-1: Discrete cable length metric states in first-stage backward; left: random seeds, 6 stations, right: VLA-like seeds, 27 stations; both cable length and uv density are normalized by the mean of the initial population; the initial population is plotted with black circles, the optimized (evolved) population is plotted with red stars; improvement upon the original population can only be detected for the 27 stations case

This is why all forward Pareto fronts are continuous and show advancement, more pronounced for large number of stations.

A notable difference between the two strategies is the direction of motion of the Pareto front. As seen in Figure 3-2 the backward population moves to the left, in direction of decreasing cable length, without achieving much improvement in coverage. The forward population shifts downward, in direction of decreasing uv density metric, without much decrease in cost. This means that adding new infrastructure will always come at a cost, but if done well, will improve performance significantly. The backward case is a good example of a strategy with no flexibility. Once the stations have been placed, coverage can never improve significantly, because usually subsets that span the design space well are not connected and hence are not a valid choice for the algorithm.

More plots of advancing Pareto fronts can be found in Figures A-11 through A-20.

3.3.2 Convergence

Convergence is closely related to the objective optimized for and the initial geometries seeded in the algorithm. We explore three aspects of convergence: convergence of the energy function and the two metrics, convergence as a function of geometric seeds and finally differences between the convergence of the backward and the forward techniques.

The cable length metric experiences much more variation from the population mean than the uv density metric because it is more sensitive to perturbations in the design vector. The rotational symmetry of the uv density results in a lot of different geometries having the same or similar coverage, and therefore not varying greatly. As a consequence, low energy states can be obtained by better exploiting linkage rather than coverage (the energy function is a sum of the two metrics). As observed for all cases, if the cable length converges, the total energy converges to a stable value also. In most cases where the energy converges, the uv density also reaches a lower stable value, except in the case of VLA-like geometries. Since these designs are highly minimized for cable length, the overall energy low state is found at the expense of a higher uv density metric. This is illustrated on Figure 3-3 where the nadir point convergence history for VLA-like seeded designs is shown. Both energy and metrics converge to a stable point, but the uv density increases, while the cable length metric compensates for the energy to decrease. To summarize, the cable length metric is the driving factor for energy convergence, which in itself depends on the connectivity and flexibility of the array.

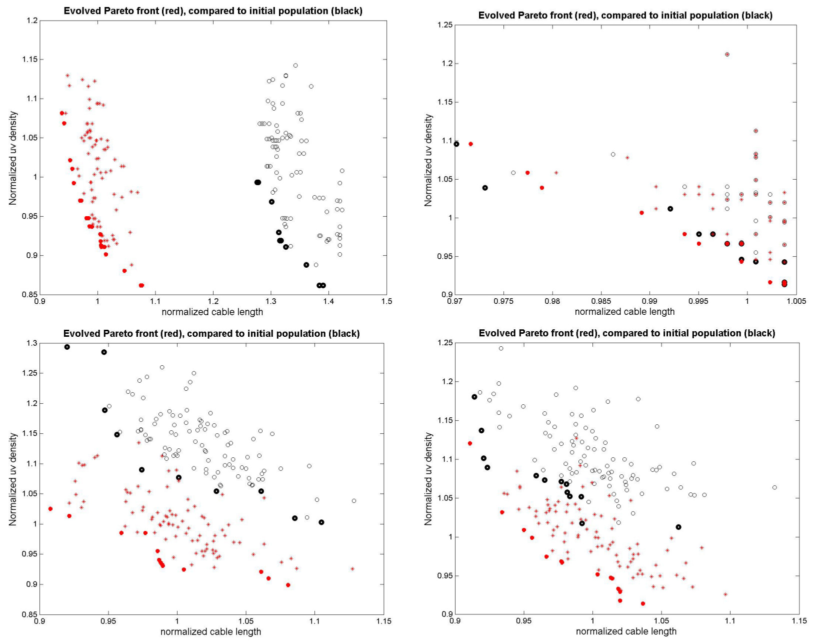


Figure 3-2: Evolving Pareto fronts, triangular seeds; top left: backward 2^{nd} stage, 60 stations; top right: backward 1^{st} stage, 27 stations; bottom left: forward 2^{nd} stage, 60 stations; bottom right: forward 3^{rd} stage, 99 stations; original populations are shown with black circles, the optimized arrays are shown with red stars

The *uv* density metric is more geometry-dependent and harder to converge.

An unexpected result is that convergence is also a function of the geometry of the array. Of the ones tested, circular, random and Reuleaux triangular always converge, whereas triangular and VLA-like do not, as seen in Figure 3-4, (especially for small arrays obtained with the backward staging strategy). Due to the smaller design space and connectivity constraint, the backward strategy exhibits poorer convergence in most cases. Improvement is sought upon a fixed infrastructure, with a rigid geometric base. If the relative position of stations is fixed beforehand, not much can be done to improve coverage. The algorithm usually then finds the least costly configuration. The forward staging technique provides more design freedom. The design space consists of randomly generated stations strewn around the design space in addition to the previously optimized "legacy" stations. The randomness allows a break from the regular geometries' tendency to be better optimized for one metric. Thus forward staging innately contains the option to improve both metrics and augment the baseline configuration, as opposed to be constrained by it.

3.3.3 Multiobjective Optimization Techniques Discussion

At the beginning of this study, three algorithms were tested for fitness for this problem. A gradient-based technique, a simulated annealing toolbox and a custom-written SA were compared against a randomly generated population of designs to access their relative performance. These tests were executed for both forward and backward cases with different number of stations and different number of generations and convergence criteria. It was quickly established that the gradient approach makes no improvement in the population fitness. As expected, gradient techniques get trapped in local minima which in a highly-nonlinear design space with rugged peaks cannot reach any sensible results. Of the two simulated annealing algorithms, the custom-written performs better, conceivably because it is better tuned to the problem at hand. All in-depth studies were performed using this simulated annealing algorithm.

For each staging principle, backward and forward, five test cases were run, with five different seed

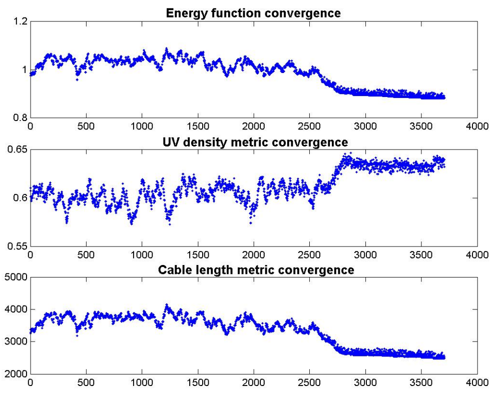


Figure 3-3: Energy and metrics convergence for a third-step forward SA with VLA-like seeds; the first plot shows energy convergence, the second shows convergence of the *uv* density function, the third shows the cable length metric convergence

Table 3.1: Table of Computational Times [min]: Backward Path, 6-15-24 number of stations

Stages	Cir	VLA	Reu	Tri	Ran
SA1	23.2 min	78.4 min	81.0 min	89.9 min	79.8 min
SA2	77.9 min	143.8 min	148.8 min	155.7 min	153.4 min
GA3	11.6 min	11.2 min	11.2 min	11.0 min	11.8 min
Total	1.88 hrs	3.89 hrs	4.02 hrs	4.28 hrs	4.08 hrs

geometries: random, triangular, circular, VLA-like and Reuleaux triangle. Each set of five cases was ran with a different set of number of stations and simulated annealing parameters. In this chapter we discuss low-number-of-stations cases with 6, 15 and 24 stations and high-number-of-stations cases with 27, 60 and 99 stations. Depending on the size of the problem, the convergence criteria were varied to match computational resources. Tables 3.1, 3.3, 3.4 and 3.2 include computation times for backward and forward techniques for different number of stations per stage. In general, lower number of stations per stage reduce greatly the computation time.

Another general trend is that backward staging takes generally less time than forward. The reason is that simulated annealing is the more time-consuming algorithm for this problem compared to the first-stage GA. The backward GA is slower than the forward GA since in the backward path, the GA optimizes the last stage (greatest number of stages), while for the forward, the GA optimizes the smallest number of stations (first-stage). The simulated annealing step, on the other hand, (which is more time-consuming per iteration), handles much larger sets of arrays in the forward path and outweighs the shorter GA.

For low numbers of stations, the backward strategy is actually slower, primarily because of its DOE algorithm B.3 which is considerably harder than the forward DOE B.4. Computing connected subsets is very slow because iterations over all the nodes and their neighboring nodes are needed. So, for smaller arrays, the DOE computation time compensates for the smaller design space of the backward strategy and makes it relatively slower than the forward.

3.3.4 Non-dominated geometries $(27 \Rightarrow 60)$

An intriguing question is what Pareto-front configurations look like. In various mathematical problems, regular geometries provide the optimal solution, and this is why it is interesting to ask the same question about optimal networks. First of all, are the Pareto designs represented by regular geometries? Secondly, we ask the question of where different regular geometries are placed along

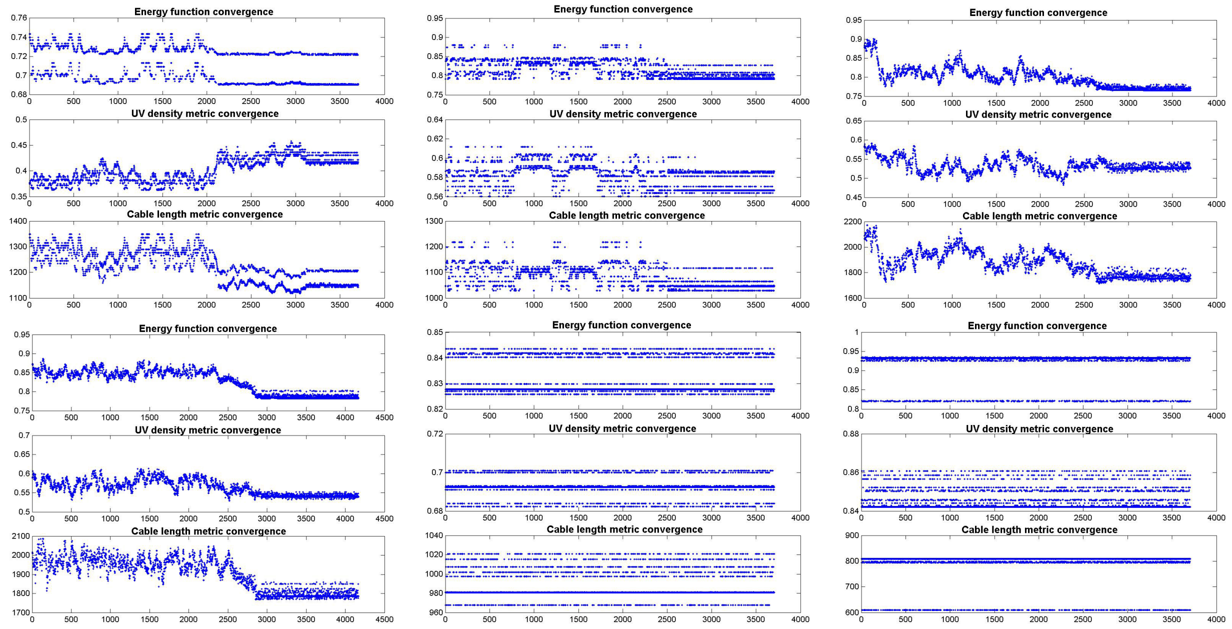


Figure 3-4: Nadir point history convergence for the backward strategy with 60 stations, for six geometries, from left to right, top to bottom: circular, Reuleaux triangular, random, log spiral, triangular and VLA-like; each plot shows the energy metric, the *uv* density and the cable length convergence from top to bottom; all geometries except for triangular and VLA-like exhibit good convergence history

Table 3.2: Table of Computational Times: Forward Path, 6-15-24 number of stations

Stages	Cir	VLA	Reu	Tri	Ran
GA1	0.93 min	1.19 min	1.65 min	1.01 min	1.69 min
SA2	38.8 min	34.8 min	33.5 min	39.9 min	37.2 min
SA3	88.1 min	98.4 min	90.5 min	89.4 min	88.2 min
Total	2.13 hrs	2.24 hrs	2.09 hrs	2.17 hrs	2.12 hrs

the Pareto front, since some might be optimized for a particular objective. For example, we expect to find circular designs to have the best coverage, that is best *uv* metric. And finally, an important question is whether the evolution of randomly seeded geometries results in a regular pattern.

In this section, we will show that the answers to these questions are different when obtained with backward and forward staging techniques, where not only some solutions are favored by one algorithm and not the other, but also new geometries emerge.

By *non-dominated* geometries we mean the geometries that represent the best trade-off between two metrics, where we choose the two metrics to be any representation of cost and performance for a set of stages. For a single-stage array the metrics are simply the cable length and *uv* density. For a Pareto front based on two stages, the two metrics from both stages are combined in a linear fashion and their combinations subjected to a Pareto filter. For all results in this section, a linear equally-weighted combination of *uv* density and cable length are used. After comparing the results from different stages, it turned out that in a three-stage optimization for example, there is a lot in common between the Pareto designs based on first-second stage and second-third. Here, we only present the results based on first and second-stage.

Backward staging tends to produce strictly regular geometries because it is based on GA-optimized highly regular designs. A good example is the 60-station case reduced from first stage

Table 3.3: Table of Computational Times: Backward Path, 27-60-99 number of stations

Stages	Cir	VLA	Reu	Tri	Ran
SA1	52.1 min	52.1 min	52 min	51.8 min	54.3 min
SA2	4.8 hrs	4.5 hrs	4.5 hrs	4.5 hrs	5.4 hrs
GA3	43.8 min	43.8 min	42.1 min	43.5 min	135.6 min
Total	6.4 hrs	6.04 hrs	6.07 hrs	6.12 hrs	8.55 hrs

Table 3.4: Table of Computational Times: Forward Path, 27-60-99 number of stations

Stages	Cir	VLA	Reu	Tri	Ran
GA1	3.2 min	3.4 min	3.2 min	3.3 min	3.5 min
SA2	6.17 hrs	6.03 hrs	6.24 hrs	6.10 hrs	6.34 hrs
SA3	21.6 hrs	20.96 hrs	21.91 hrs	21.4 hrs	22.0 hrs
Total	27.8 hrs	27.05 hrs	28.2 hrs	27.6 hrs	28.4 hrs

99 for circular and Reuleaux triangular seeds shown in Figure 3-5. This also explains the lack of

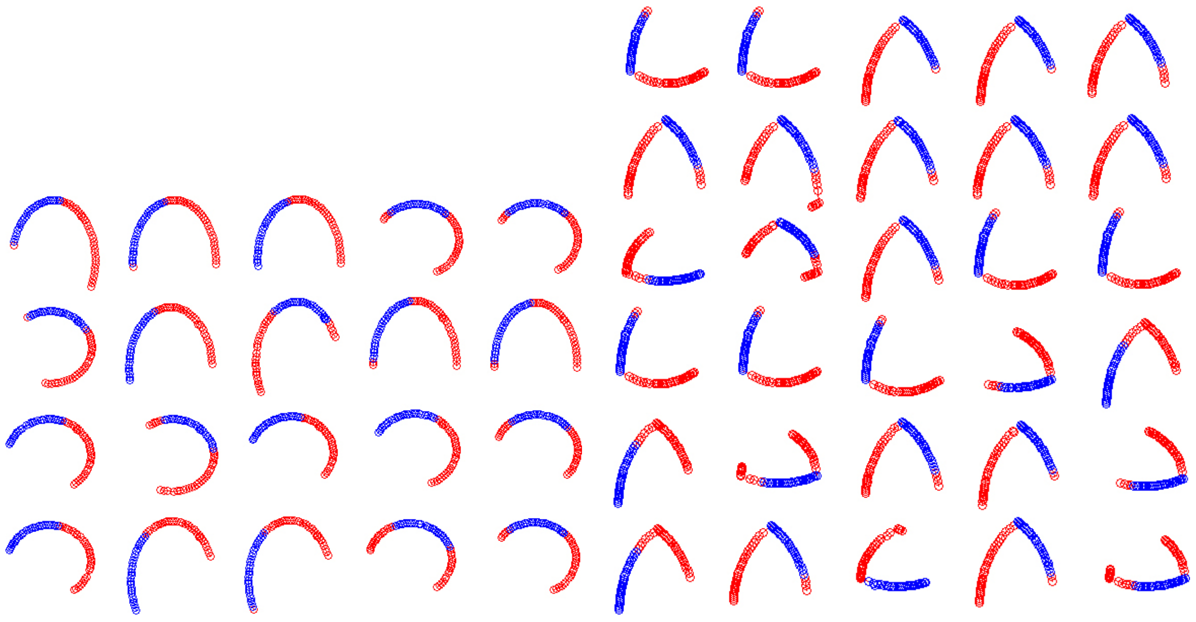


Figure 3-5: Non-dominated geometries; backward strategy; 60 stations, circular seeds (left) and Reuleaux triangular seeds (right); the blue segments indicate the selected optimal 27-station first-stage subset

significant improvement (convergence) of designs. Starting from a nearly optimal condition, a reductionist approach moves away from the global optimum, especially in coverage. Cable length changes are also constrained because the geometry is fixed. Thus the performance of the backward strategy degrades as the stage number decreases.

In the forward case, the regular geometry is less prevalent. The base configuration is used as a building block for protruding branches that increase coverage with minimal possible length span. As seen in Figure 3-6, for spread out configurations like circular shapes and Reuleaux triangles, relatively short protruding branches extend inward, whereas for more compact geometries like the VLA-like, the extending branches radiate out to fill up uv space.

Random-seeded array geometry evolution reveals the nature of the different optimization principles. As designs move closer to the Pareto front, random geometries assume regular patterns

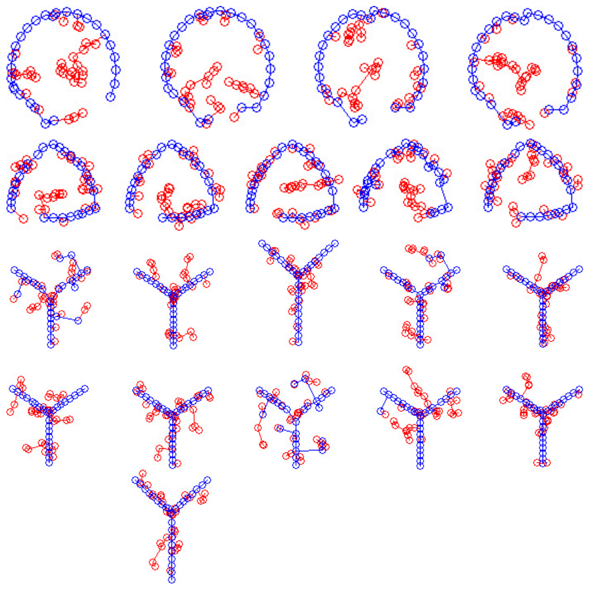


Figure 3-6: Non-dominated geometries; forward strategy; 60 stations; circular seeds top, Reuleaux triangular seeds middle and VLA-like bottom; the red circles indicate the second-stage additions to the blue "legacy" arrays

though never fully. A good question is whether an absolute optimum geometry can have a non-regular configuration, semi-regular or has to be geometrically perfect. A complete answer can only be given after extensive simulation which might suggest a theoretical argument. Our results suggest that backward and forward staging provide very different answers. Figure 3-7 shows random seeds evolved to Pareto geometries for backward (left) and forward (right) strategies. It is evident that the

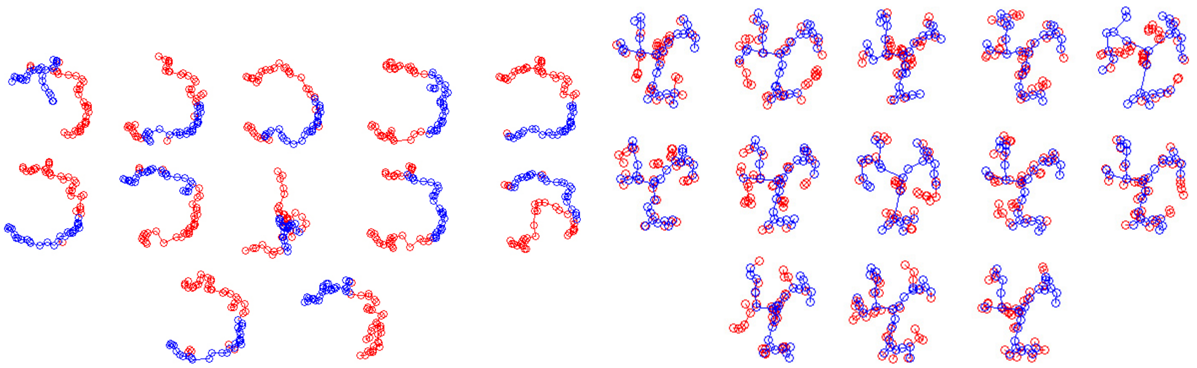


Figure 3-7: Non-dominated geometries with random seeds, left: backward strategy, right: forward; the blue indicates the selected optimal 27-station first stage; backward strategy geometries resemble incomplete circles, while forward-obtained arrays approach a log spiral geometry

backward geometries converge to incomplete circles, while the forward results reveal a new shape - spiral arms reaching out in a VLA-like manner. An cartoon representation is given in Figure 3-8. This new semi-regular shape resembles a hybrid of a VLA-like and circular geometries. While minimizing the cable spread, by curving the arms of the VLA, the optimizer improves the coverage. This result provoked interest in *log spiral* geometries which have been proposed previously for telescope arrays [14]. Simulation experiments were performed for both backward and forward strategy only to confirm that log spiral seeds move in objective space in a similar manner to random geometries, for both backward and forward strategies. Figure 3-9 shows how the non-dominated geometries of log spiral seeded population look very much like the random results. Our preliminary results also show that log spirals do not supersede the rest of the tested regular geometries and are not Pareto optimal most of the time. This is further discussed below in the comparison between different seeds in objective space.

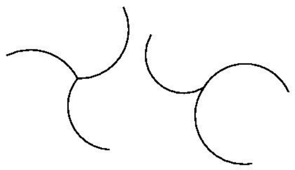


Figure 3-8: Cartoon representation of the non-dominated random seeded forward geometries' resemblance to log spirals

Another interesting question is where the different geometries reside on the Pareto front. Our results confirm Cohanim's investigation [12] which shows that circular geometries are best optimized for the *uv* metric, VLA-like for the cable metric, so they assume the two corners of the Pareto front and Reuleaux triangular shapes occupy the middle part. Random and log spiral geometries often remain sub-optimal. An example is given in Figure 3-10 which shows second-stage results of Pareto fronts for different geometries (circular seeds are plotted with circles, VLA-like with pluses, Reuleaux triangular with diamonds, triangular with triangles, random with crosses and log spirals with stars). On the left plot, circular seeds are the designs with the smallest *uv* metric. The VLA-like (top, left) have the best cable length metric, Reuleaux triangular are a good compromise between the two, while random, triangular and log-spiral are consistently behind the Pareto front. This behavior is similar in the forward strategy plot on Figure 3-10 (right).

All non-dominated geometries, for all geometric seeds, for both backward and forward approaches with different number of stations are presented in Section A.3.

3.4 Backward Versus Forward Staging Principles

The key difference in the two staging techniques is the direction of optimization in time. The forward technique starts with a GA-optimal design and augments it using an optimizer to construct the following stages. The backward technique is based on optimality in the future, not the present, and starts with a GA-optimal design (envisioned with today's metrics (goals)) and reduces it using the same optimizer as the forward strategy to deduce the previous stage. Both techniques address the question of staged deployment, but have diametrically opposite optimality priorities. The question addressed in this section is which principle, if any, produces better results, and under what conditions.

- Backward focuses on optimality in the future, and makes the best of the present based on its future vision. Forward focuses on optimality in the present and builds upon it to create future designs. As designed, backward is more deterministic, while forward features more flexibility with its greater degree of randomness. An example of the superiority of backward in the last stage is provided by Figure **??** (right).

- Backward works with a static predetermined geometry, and hence produces results which are reduced versions of Pareto-optimal designs. This makes the backward strategy almost predictable. Forward chooses new designs from a randomly generated set in the design space, thus providing the opportunity for the emergence of new geometries. To claim that results converge to regular patterns in high stages in the forward strategy, a lot of experiments need to be made and analyzed statistically. Thus the forward strategy is much less predictable than the backward approach, though certain behavior can be deduced from the goals of the optimizer.

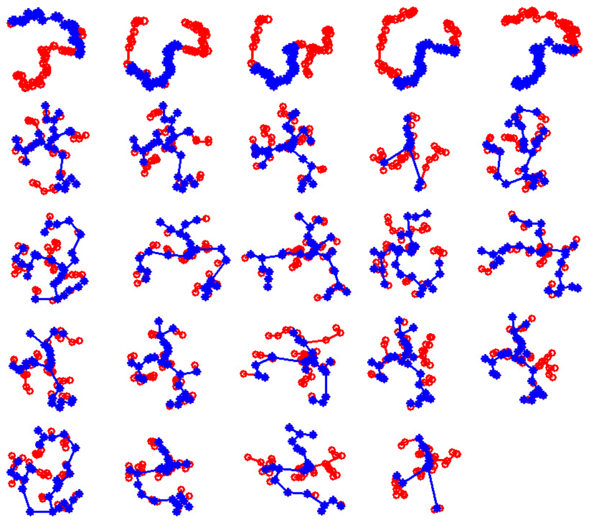


Figure 3-9: All non-dominated log spiral geometries; backward (top 5) and forward (rest); this figure shows that non-dominated log spirals for both backward and forward strategy resemble the random non-dominated seeds

- Even with similar design space sizes, backward and forward do not show the same convergence and Pareto front advancement. All cases reach stable solution points in the forward case: all geometries, all stages for all numbers of stations. In the backward case, not all non-dominated geometries show consistent convergence history, but instead stable oscillations between few discrete energy states. The primary reason for this is that the backward design space is more constrained, the stations' positions and their connectivity are fixed from the beginning. Therefore, the optimal subset of an initial bigger array can neither have a radically different geometry, nor connectivity. Among all the subsets, especially for regular geometries, there is a lot of similarity. This is why the convergence histories show few energy states that the system oscillates between. In the case of forward, there are many more degrees of freedom. First, new station positions are allowed, and with them renewed connectivity. It turns out that any amount of recabling (creating new links, and ignoring the old infrastructure) provides more flexibility in the design. Complete recabling (also in the case of backward) has not been considered in this analysis because it is believed to be very costly. In a strictly mathematical framework, though, recabling will definitely improve the performance of the backward strategy.

- Per stage comparison confirms that backward and forward are best for different stages, but also reveals consistent patterns in their relative performance. The first-stage results for backward and forward (27 stations) are compared in Figure 3-11 (left). The first-stage forward is optimized with a GA algorithm, which results in a balanced set of designs with satisfactory performance and best possible cable lengths for that performance. The backward results for 27 stations are a second-time SA optimization of optimum 99-station designs. They are naturally low-cost (lowest cable length), but because of the reductionist philosophy of backward, they also have poor performance. The second stage comparison, given for VLA-like seeds in Figure 3-11 (middle), invokes similar conclusions, though the differences are not as striking as in the first-stage case. This is the passing (intermediate) stage between the domination stages for each strategy. Another viewpoint of the per stage comparison is an assessment of the best each strategy can do at every stage. Results are shown in Figure 3-12. At its best, the forward strategy is always more costly (Figure 3-12 right), but it outperforms the backward strategy in the first-stage (Figure 3-12 middle).

  In summary, the forward is always the more expensive approach and the better performer, except in the case of third stage, where the backward strategy is the winner in both objectives.

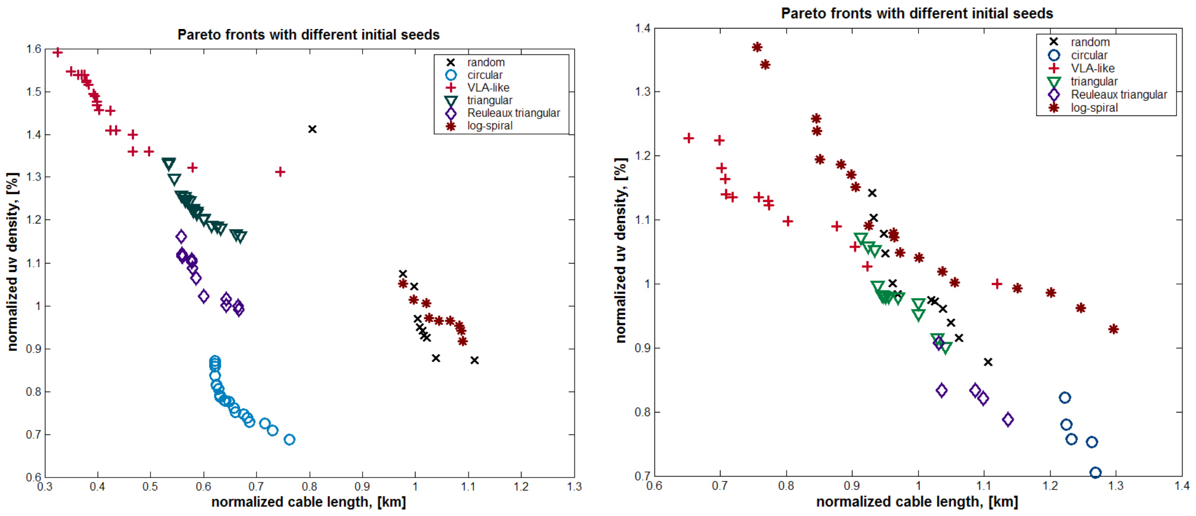


Figure 3-10: Second-stage backward (left) and forward (right), 60 stations, Pareto fronts for different geometric seeds: VLA-like (pluses), triangular (triangles), Reuleaux triangular (diamonds), circular (circles), random (crosses) and log spiral (stars)

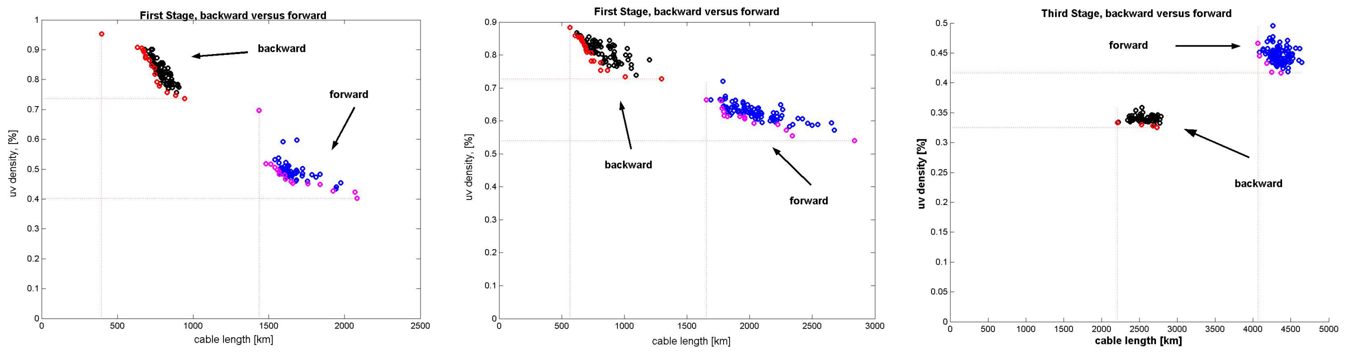


Figure 3-11: Backward versus forward comparison per stage; left: first stage, 27 stations with random seeds, middle: second stage, 60 stations with VLA-like seeds, right: third stage, 99 stations with circular seeds

- The results presented in this chapter show that there are many more fundamental differences between the two staging principles chosen than believed at first. They turned out to be completely different approaches at tackling the same problem, with different convergence times and distinct results. Our hypothesis that the backward strategy will always be better overall has clearly been disproved. The superiority of backward in the third stage, cannot outweigh its poor performance in the early stages. The lesson learned is that backward is the strategy to follow only if the goal is to construct intermediate steps or building blocks to reach a final vital goal. In an uncertain environment, with a lot of intermediate goals, and desired sustained system performance at all times, forward is the strategy to take. Forward not only allows adaptation to changing requirements, like budget or policy constraints, but it also allows attaching priorities to different stages.

3.5 Upfront *Penalty* of Scalability in Stage 1

Scalability saves infrastructure cost, reconfiguration time, and as this thesis claims provides the overall best pathway to staged design. However, like built-in functionality or flexibility, extensibility

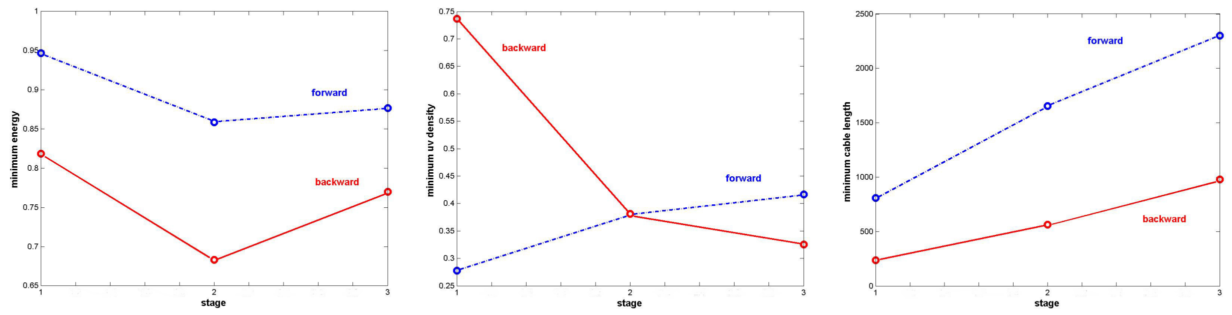


Figure 3-12: Lowest achievable energy per state for backward and forward strategies (left); minimum *uv* density metric (middle) and minimum cable length (right); evidently forward staging always costs more than backward staging, but in terms of coverage the two methods are superior in different stages

comes with an upfront cost. Minimizing this upfront investment is part of the design procedure, and for most problems it cannot be eliminated, especially using the same design techniques, current technological knowledge, and no insider's knowledge of the future. For telescope arrays, the penalty concept is simple. If the array is to be designed to near-optimality in the future, then it cannot be fully optimal now. Any addition to and extension of its configuration will then produce a less optimal design. Following this intuition, others have also claimed before [4] that Pareto optimality cannot be achieved on the same design path, for all design stages. According to [4], the optimal staged path, follows closely the Pareto front, never completely approaching it.

To test the above assumptions, here we look at the best (*nadir-best*) results achieved by both backward and forward simulated annealing techniques and compare those, for each stage, with the benchmark GA algorithm. Table 3.5 contains the metrics (or coordinates) of nadir points for all stages, for both backward and forward. The GA equivalent result is given in the last column. For each stage and principle, the nadir-point design is given by the pair ($uv\ density, cable\ length$) and the seeded geometry of that particular design is shown by the index. Starting with the stage 3 results for 99 stations, we find no surprises. The backward 3^{rd} stage, which is a GA routine outperforms the forward, since it is a single-point design, dedicated to 99 stations. It has about half the cable length than forward and about 40% better coverage. In the second stage, for 60 stations, both backward and forward excel in one metric only. Forward has better coverage (smaller *uv* density), but twice the cable length. A winner cannot be singled out without further considerations or changing the weight on the metrics. The GA results for the second stage outperform both the backward and forward SA algorithms. The first stage results are slightly different from the expected. Since

Table 3.5: Table of best designs' coordinates ($uv\ density$, $cable\ length$) per stage; index indicates geometry; a GA single-stage optimization result is provided for benchmarking

Stage	Backward	Forward	GA
1^{st} – 27 stations	$(0.9772, 235.8)_{VLA}$	$(0.3248, 1545.6)_{REU}$	$(0.3248, 1545.6)_{REU}$
2^{nd} – 60 stations	$(0.5667, 1045)_{REU}$	$(0.4508, 2626.4)_{REU}$	$(0.3472, 1015.8)_{REU}$
3^{rd} – 99 stations	$(0.3705, 1733.2)_{REU}$	$(0.5230, 3339.1)_{TRI}$	$(0.3705, 1733.2)_{REU}$

forward uses a GA algorithm to optimize this stage, it is expected that it will outperform backward on both metrics. Due to the nature of the backward algorithm though, most designs degrade very heavily in coverage for the benefit of having very low cable length. This makes the cable length of backward for 27 stations about 6 times smaller than that of forward. From a scientific point of view though, a *uv* density higher than 0.9 cannot be acceptable. To meet scientific objectives, we cannot accept the backward first stage as a good design. As mentioned earlier, recabling is a method that can improve the outlook of backward designs in the first and second stage. This means that when subset configurations of stations are considered for optimization, their connectivity is overlooked, but instead, they are linked anew in an optimal (minimum spanning tree) manner.

Our arguments indicate that penalties for staging must be paid sooner or later in the design process. If designing for an optimal future stage, one has to allow non-optimality in the prior designs to allow for the restructuring and reconfiguration to lead to better results. Extensible in this context means *allowing changes to improve the design.* This will involve including features that do not necessarily benefit the present configuration or might not even be needed. On the other hand, if designing with an optimal start, one has to always deal with the rigid infrastructure laid down at the beginning. Extensions and additions to an already optimal design usually decrease its optimality. As seen, designing forward might well target performance and deliver satisfactorily, but it inevitably leads to increasing cost and limited asymptotic performance. Designing backward will most likely turn out to be cheaper, but not necessarily satisfy the value objectives of the endeavor, particularly during early stages.

3.6 Lifecycle Benefits of Staging Optimization

This section focuses on the long-term benefits of staging. Redesigning large engineering systems during their lifetime is a complicated process which involves partial or substantial replacement, extension of existing parts, reconfiguration and adaptation. All of these processes may be possible, difficult or inconceivable depending on the state of the system and the conditions it has to operate under. We claim that making scalability and evolution part of the design process, makes not only operation easier and performance better, but also reduces overall cost.

In traditional design a system is designed with little flexibility to accommodate changes in configuration and functionality. The systems that survived time were either those simple and elegant enough to never lose functionality and need replacement, or those that discovered new functions and adapted quickly. Despite the upfront penalty, designing systems to adapt can not only increase their lifetime, but also benefit greatly the user. In the context of telescope arrays, a complete replacement and re-design of the system means that for every new set of stations, the array is optimized anew, with new physical locations and new links. This can be extremely expensive, given the cost and availability of digital optical fibre. Some websites list digital optical fibre for home appliances at around \$50 per meter [17]. This quickly amounts to 50000 USD for a kilometer and 150 million USD for 3000 km, a distance envisioned to connect some of the ground-based telescopes in Europe.

This case is a good example where staged deployment and scalability should be inherent to the initial design. To address this question, we looked at the best designs based on two and three stages (combined metrics) together with the single-stage winner. Table 3.6 complements the numbers presented in Table 3.5 by showing the nadir-point solutions when the metrics of two $(1, 2), (2, 3)$ or all three stages are combined linearly with equal weights. Clearly, those are different from the single-stage winners, which confirms the fact that best design paths do not necessarily lie on the Pareto front. In terms of geometries, the Reuleaux triangular shapes are usually among the nadir-points, however sometimes another geometry fares better through design space, like triangles. Circular or VLA-like patters consistently populate the anchor parts of the objectives space. Figure 3-13 shows

Table 3.6: Table of best second-stage (60 stations) designs' coordinates (*uv density*, *cable length*) based on combining the objective of two (or three) stages; index indicates geometry

Stages	Backward	Forward
$1, 2 - 60$ stations	$(0.8831, 564.1)_{VLA}$	$(0.4508, 2626.4)_{REU}$
$2, 3 - 60$ stations	$(0.5912, 1025.3)_{REU}$	$(0.5302, 2410.3)_{TRI}$
$1, 2, 3 - 60$ stations	$(0.6186, 971.8)_{REU}$	$(0.4619, 2668.8)_{REU}$

the entire populations with their (darker) Pareto fronts for stages with 27, 60 and 99 stations. On both plots, the grey population corresponds to first stage, the blue to second and the green to third. Nadir points for each population are plotted in yellow. For example, in the backward plot, the

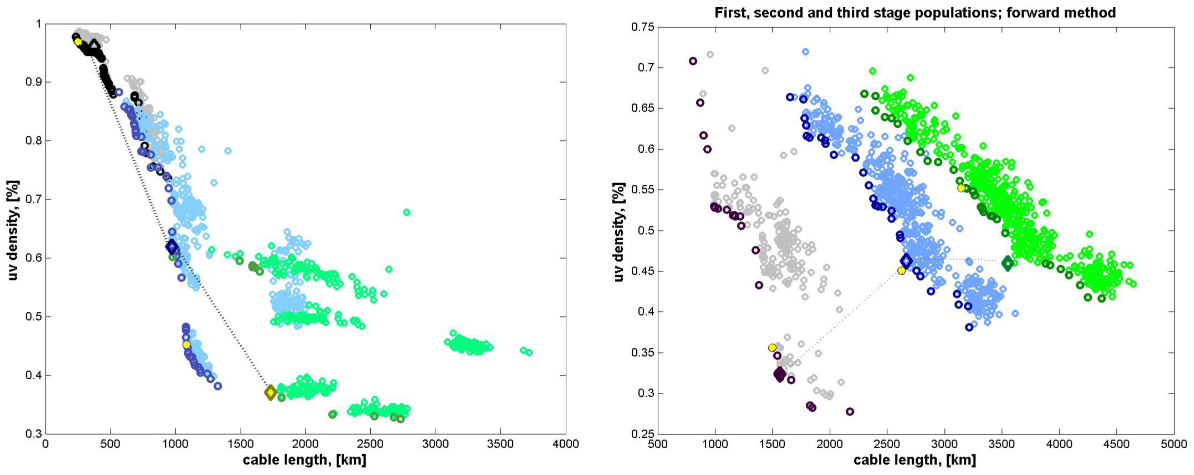


Figure 3-13: Best design paths across three stages; left: backward, right: forward; first stage (27 stations) shown in grey, second stage (60 stations), shown in blue and third stage (99 stations) in green; the nadir-utopia point for each population is marked with yellow; the best design paths are shown with diamonds connected with dotted lines

third stage solution starts out as excellent performance $(0.3 - 0.4)$, low cost $(1500 - 2000$ km for 99 stations), then evolves to a good performance (0.4-0.5) and reasonable cost $(1000 - 1100$ km for 60 stations) solution, to finally end at an extremely bad performance and extremely low cost solution in the first stage. The backward Pareto plots and nadir-points in Figure 3-13 (left) show that with the backward progressive optimization, starting with a GA routine and then two consecutive SA algorithms, the array cost does not vary much, but remains low. The *uv* density slope on the other hand is very steep, to show that performance decreases drastically as stations are removed. This is a direct consequence of keeping the linking (connectivity) of the array and thus keeping cost low. Due to inevitable trade-off, the performance plummets down.

In the forward staging plot on Figure 3-13 (right) the populations evolve more uniformly with both metrics increasing at every stage. Each consecutive nadir point (in yellow) has increased *uv* density metric and increased cost. Since the first stage in forward is GA-optimized and considered the most optimal, it is clear that with every new SA-optimization, the algorithm never reaches the performance of the initial array. On the other hand, forward never shows plummeting coverage either. The third-stage nadir point still has a *uv* metric around $0.55 - 0.6$ which is still better than the second-stage backward best point.

Regardless of the position of the nadir points across designs, they do not represent the time evolution of the same array. Choosing the nadir points to build at each stage can turn out to be prohibitively expensive due to reconfiguration and relinking, as discussed above. Instead, we look at the *best design paths*, by using the linear combination of metrics for the three stages to choose the nadir-point. These arrays in the populations are marked with diamond shapes in Figure 3-13 and connected with dotted lines to show the design paths. It is interesting to see that sometimes these points are very close to the nadir-points of the population, and even could coincide with them, as in the case of backward, third stage, but they can never overlap completely. The other interesting observation is that the best design points usually are on the Pareto front for their own stage, even though they do not coincide with the nadir-point. For the designer this means that even for configurations where the best metrics are not selected, the best trade-off is.

The best design path solutions, for both backward and forward are shown in Figure 3-14.

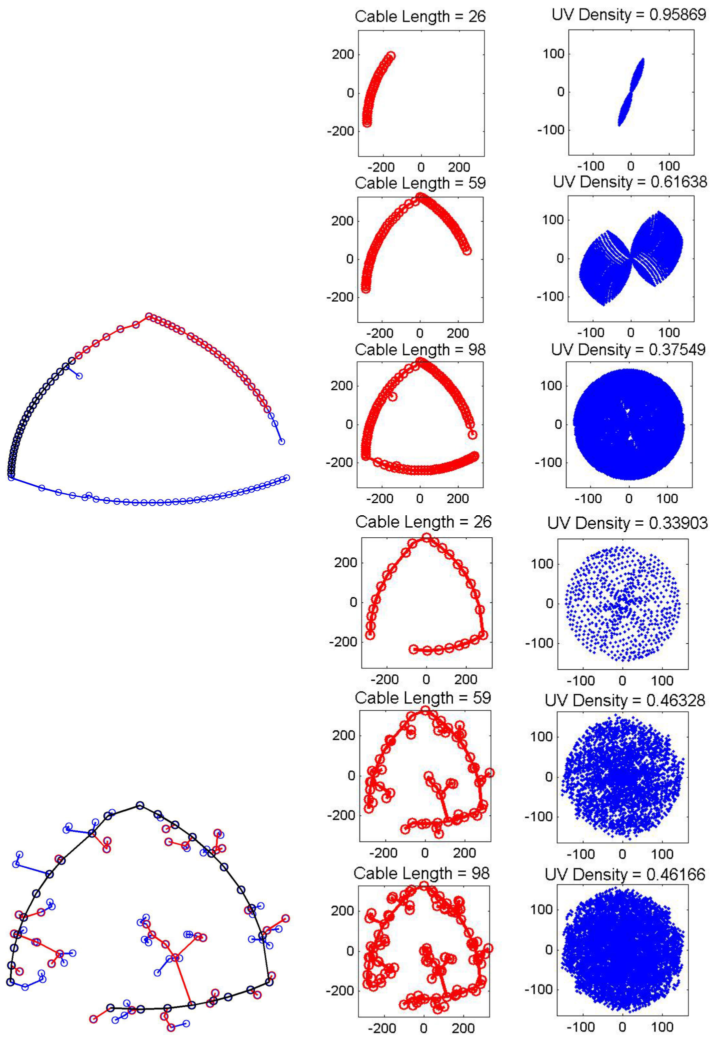


Figure 3-14: Best path configurations with their uv density, based on metrics from three stages ($27 \rightarrow 60 \rightarrow 99$ stations); top: backward, bottom: forward

Chapter 4

Network Robustness

The study of systems has changed drastically in the past hundred years. Fundamental sciences, both theoretically and experimentally have mastered the structural and functional understanding of system components by method of elements breakdown. Traditional engineering design has followed a similar single-entity focused philosophy. Systematic catalogues of physical and abstract properties have been created of organisms, natural phenomena; point design methods and refined frameworks have been created to solve well-posed specific problems. However, recent advances in complex network theory have demonstrated that large databases of collected knowledge about systems are not enough to understand their structure and functionality. A key challenge nowadays is to understand the complex web of interactions, which not only define network structure, but also determine functionality. Often elements with distinct individual functions are combined together to perform a completely different task.

Various network models have been proposed, all suitable for different classes of networks. As global interconnectivity rises, in most domains of life, it becomes more and more crucial to understand the topology and constituency of the relations governing our life, from social networking to the metabolic pathways in the cell. Transportation and financial network growth, for example, has improved life quality on a global level. Unfortunately, this rising connectivity has an inherent vulnerability. The loophole is that destroying the entire system can be aimed at single, well-connected or crucially connected units. Evidence of this is rampant throughout biology, as evolution explores niches with different species. Some examples are disease spreading through social, or environmental, or physical networks; virus exploring vital reactions and affecting vital compounds thus disabling normal processes in organisms. Designing or repairing systems in an uncertain and variable environment requires understanding the underlying network properties that determine the response of the network functionality to disturbances of its constituent parts.

This chapter defines several generic network properties and models, employed to study various natural and artificial networks and considers the problem of designing large telescope arrays for robustness versus performance and cost.

4.1 Some Background, Random versus Deliberate Attacks

The studies on robustness and network growth have become very popular in the past decade, mainly because of the increasing availability of large datasets of complex interacting systems, like the Internet or the genomes of various species. It has been claimed [18] that a lot of large complex systems show organic-like growth and exhibit a lot of emergent phenomena, that is empirical discoveries that are surprisingly derived from experiments, but cannot be deduced by simple modeling of the actual system. Log spiral geometries are an example of independently discovered regular geometries for the telescope arrays, result of many simulations, that have in fact been considered in other studies [14]. It is not mathematically clear why random seeds evolve to organize themselves as log spirals. Such discoveries have lead to extensive studies in understanding natural systems network behavior [19] to look for rigorous mathematical models, emergent behavior and how new functionality appears with the goal to "reverse engineer" manmade systems. The primary goal behind this endeavor is to hope for an insight from nature about designing well-performing, yet resilient systems.

Following the "reverse engineering" philosophy, in this chapter, we take general network models and defined metrics and attempt to redesign telescope arrays for robustness while keeping track of traditional engineering design goals.

4.1.1 Robustness Metric

In general terms, *robustness* is defined as the system's ability to respond to changes in the external conditions or internal organization while maintaining relatively normal behavior. *Topological robustness* refers to the fact that disabling a substantial number of network nodes results in inevitable functional disintegration. Small disturbances can also be detrimental if targeted well. For example, if an airline hub airport is suddenly closed due to a blizzard, most of the airline's flights will not be able to follow their normal itinerary, will not be able to fly at all, or will have to be rerouted. *Functional and dynamical robustness* reflects changes in the network dynamics and inherent functions due to environmental or internal disturbances. In the airline example, this corresponds to the inability of the airline to transport passengers, or serve certain destinations. Loss in function can result in loss of market share and thus revenue (or profit, or both) loss. This is why understanding network class types and their relative robustness is essential to designing networks and preventing failure.

There are examples of incredibly resilient networks (Internet) which still form connected components and perform vital functions, even if a significant portion of their nodes are removed. Scale-free networks exhibit this behavior, even when 80% their nodes are removed, the remaining 20% still form a connected component with a path linking every node to every other node. Evidence shows [19] that in *S.cerevisiae* (yeast) only proteins with more than 15 interactions are essential. This means that connectivity has an important role in the deletion phenotype.

First, we concentrate on topological robustness. Functional robustness, as applied to telescope arrays, is obtained by the same method. We look at disturbances caused by two types of failure - node or edge removal (malfunction). The structural (topological) robustness is defined as the percentage of nodes still connected to the hub (via some path) after the removal, where the hub is defined as the highest-degree (most connected) node. A node is connected to the hub if its shortest path to the hub is not infinite. The shortest path algorithm is described in section 1.3.2. Figure 4-1 show how the robustness metric is calculated. Notice that such a metric can be calculated once, for a randomly hit node (or group of nodes) of the graph or averaged across the whole network. The *average robustness metric* is the average of the metrics computed by knocking out systematically all nodes (and/or edges) of the graph.

Even from a simple scenario as described in Figure 4-1 it can be seen that there is a significant difference between random and deliberate attacks. Depending on the network topology, a random attack can be mostly harmless. For scale-free or hierarchical networks, the majority of the nodes have a low-degree, hence are connected to very few other nodes. A random hit most likely will remove a single end-node which will not affect the flow through the network much and will not result in loss of too much information. The case might be different for random networks, but those are not met too often in biology, social sciences or engineering. A deliberate attack on the other hand, is likely to target a hub, and hubs are usually easily detected even in a non-explicit network because all roads lead to them. Clearly, it is advisable to design networks with a multi-hub architecture, especially if they affect many entities (transportation, financial, social networks). An example of a highly robust, multi-hub, hierarchical network is the Internet. The goal of this chapter is to establish whether any of the above recommendations can improve telescope arrays robustness and whether robustness is an opposing objective to cost and performance.

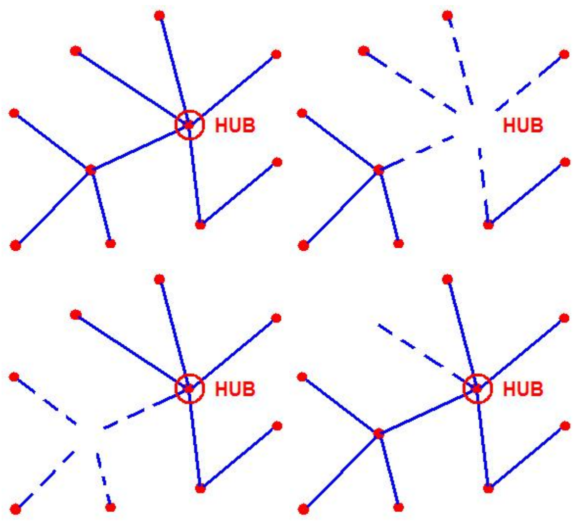


Figure 4-1: Robustness metric calculation method; 1) hub - the most connected node; 2) knocking out the hub, 0 stations connected to the hub, robustness metric is 0; 3) knocking out a highly-connected node, 6 out of 10 stations connected to the hub, robustness metric 0.6; 3) knocking out a low-degree node; 9 out of 10 stations still connected to the hub, robustness metric 0.9

4.2 Post-processing, telescope array robustness (theoretical framework)

For the telescope arrays design evaluation, the robustness metric is defined as explained in Figure 4-1. For each array, a node and a vertex robustness are calculated separately and then combined in an equally weighted sum to calculate the total robustness. The node robustness metric calculation involves removing a given node from the network and then counting the fraction of nodes still functional and connected to the hub (the *hub* is defined as the most highly connected node and might vary from array to array). The edge robustness metric is calculated with an equivalent operation, where edges are removed, instead of nodes. Certainly, the higher the fraction of surviving stations, the bigger the size of the remaining connected component, the more robust the array is to a single node-edge failure. Thus the higher the robustness metric, the more resilient the network is. For all results presented in following sections, the *average robustness metric* was used, which is an average of the fractions of stations remaining connected to the hub when all nodes are removed consecutively. Averaging the single-node robustness metric removes the randomness of the attack and provides a better comparison measure between different arrays.

This robustness metric, based on node and edge failure is a strictly topological measure. It is a function of connectivity only, not of geometry, and so differentiates between robust and non-robust linkage.

Functional (or dynamic) robustness refers to estimating the loss of function (coverage for the telescopes) due to node-edge failure. This type of robustness does reflect the geometry of the array, and so differentiates between robust and non-robust configurations. It is calculated by first computing the *uv* density of the complete array, then the *uv* density of the reduced array to obtain the percentage of lost functionality. Depending on the spread of the configuration, attacks might affect its performance differently. Dynamic and topological robustness are not independent metrics, since both are affected by connectivity. If a station is not connected, but still operating, due to edge failure, its information will not reach the hub anyway. Therefore, functional robustness accounts for both the geometry and the connectivity of the array.

As defined, both topological robustness and dynamic robustness are a number between 0 and 1, where 0 stands for minimal robustness and 1 is the maximum.

4.3 Results for the telescope array case

4.3.1 Topological robustness

Topological robustness has been evaluated for backward and forward populations for different stages. In general, the average topological robustness for all arrays is in the range 0.6-0.8. An optimal value (≈ 1) is never reached because all configurations designed in this study are minimal spanning trees, which means that there is a unique path connecting every pair of vertices. Therefore, once a station or a link is removed, all of the nodes connected to it (opposite from its path to the hub) are lost. In general graphs this doesn't have to be the case, because there may be multiple ways to get around. Thus we note that Steiner graphs are not among the most structurally robust graphs in general.

In the family of minimal spanning trees, certain expectations related to geometry also exist. Starlike geometries, where most stations are directly connected to the hub, experience minimal loss because on average, a single hit disables only one station. VLA-like configurations should be less robust because stations lie on long arms stretching out of the hubs, and depend on many other stations for connectivity. All closed geometries, like closed circles, Reuleaux triangles and triangles will not be affected at all from a single hit: on a circle, for example, removing a single node still results in a connected array. On the other hand, the linear connectivity of closed shapes, does not favor multiple attacks which can disintegrate the array structure completely. Some of these arguments do appear explicitly in the results.

Figure 4-2 and Figure 4-3 show the robustness metric estimated for second-stage backward and a second-stage forward with 60 stations. On a plot of *uv* density versus cable length, robustness is color-coded, with light signifying more robust, and dark less robust design. Notice that this is not the absolute topological robustness plotted (a number between 0 and 1) but a relative robustness that allows to intensify the contrasts inside each population. The absolute robustness metric covers a smaller interval (0.6-0.8), therefore for plotting clarity this interval was mapped to [0 1] in order to create more contrast between more and less robust designs.

The dark and light patterns seen in Figure 4-2 correspond to different geometries. Figure 4-4, which shows the Pareto fronts of different geometries for the backward second stage, serves to map the different geometries to the robustness patterns. The dark low-robust patch in the center of Figure 4-2 corresponds to the blue Pareto front of circles on Figure 4-4. Using this color mapping, we establish that random configurations are relatively the most robust of all, with some really good designs (white squares) away from the Pareto front. VLA-like geometries (top left corner of Figure 4-2) also perform well, as expected, followed by Reuleaux triangles, triangles and down to the least robust circles. An almost identical pattern is detected on Figure 4-3. Least robust are circles (in the lower right corner), followed by Reuleaux triangles and triangles (in the middle of the Pareto front). Random geometries (entirely behind the Pareto front) and VLA-like configurations are the most robust.

4.3.2 Dynamical robustness

To reiterate, dynamical robustness is a function of both geometry and connectivity. As a result, the different topologies do not have the same rank in functional robustness as in structural. Circles look better, especially with protruding arms as in the evolved forward SA higher stages. In general, designs that start out as better performers, remain high in rank. Also, as seen on Figure 4-5 which shows results for a first stage with 27 stations, there are much wider gaps in dynamic robustness among the same population than in structural. Since dynamic robustness is dependent on both stations locations and their connectivity, this metric is more sensitive to single-node or single-edge

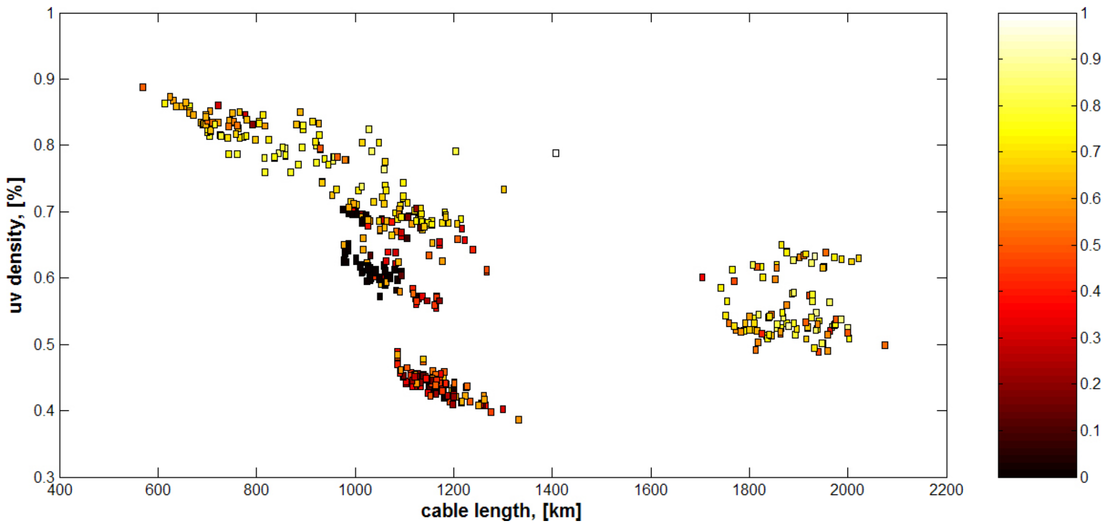


Figure 4-2: Topological robustness metric indicated by color scale for each design point in objectives space (lighter is more robust), second-stage backward strategy, all geometries, 60 stations

attacks, and hence it differentiates designs more severely.

Random-seeded configurations win in both techniques, backward and forward. To summarize, the robustness metric proves to be entirely dependent on the array topology. The same conclusions about robust versus non-robust geometries can be extracted from both backward and forward strategies results. Also, at the population level, forward obtains better robustness results, in both the structural and functional case, evident by the lighter more uniform patterns on Figures 4-2, 4-3 and 4-5.

4.3.3 Small worlds and hierarchical networks

The difference between the fitness of certain geometries can be explained using network theory. In Chapter 1 we outlined three network models developed to understand real networks: hub-centered, hub-and-spoke and a uniform-degree, where no nodes can be distinguished as hubs. According to Barabasi's [21][20] small-world model, the hub-centered, from hub-and-spoke to hierarchical networks are much more robust than uniform-degree and random graphs. Their major claims come down to a simple probability-based observation, already discussed in Section 4.1.1. Scale-free nets are characterized by an exponential degree distribution which means that the majority of nodes have low degree (are connected to very few other nodes). A random attack will likely hit a low-degree node and not cause much structural or functional damage. In uniform or decentralized networks, all attacks present an equal threat, which makes these networks less robust. To verify the small-world model, we plot the degree distributions of the most robust array and the least robust array from the forward second-stage population, with 60 stations. Figure 4-6 shows the degree distributions. The most robust design, which happens to have random seeds has an expressed exponential degree distribution verifying its scale-free nature. The least robust design has a majority of vertices of degree 2 and equal number of vertices of degrees 1 and 3. This is typical for a circular design with randomized extensions. Clearly, this array is not hub-centered, but with a uniform degree distribution. The actual geometries of these two designs are presented in Figure 4-7. The two geometric models are good representations of decentralized and hub-and-spoke models as introduced in Chapter 1, Figure 1-4.

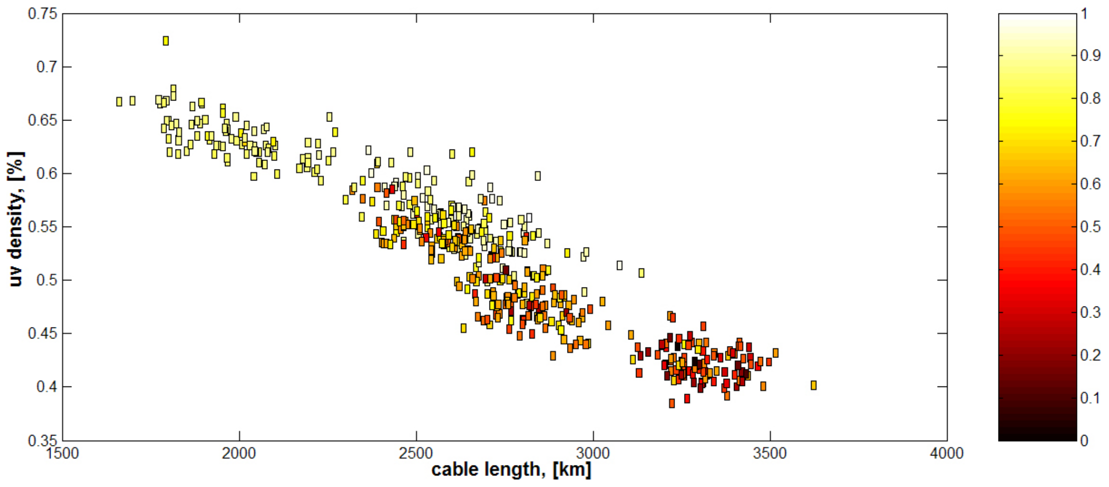


Figure 4-3: Topological robustness metric, indicated by color scale for each design point in objectives space (lighter is more robust), second-stage forward strategy, all geometries, 60 stations

4.4 Tension between Pareto optimality and Robustness

One of the primary reasons for studying the robustness of telescope arrays was the belief that optimal designs are not necessarily robust to external influences. This conjecture has not only been demonstrated but shown to be a strong statement. The intuition behind it comes the robustness and fragility principle discussed in the literature [22]. Highly-optimized systems often exhibit subtle vulnerability. In the context of graphs, this means that the network may be designed to be highly resilient to random deletion of nodes, but lose structure and functionality if targeted at specific high-degree nodes.

All results, in Figures 4-2, 4-3, and 4-5 confirm this tension between robustness and Pareto optimality. The most robust designs (color-coded in white) are usually far from the Pareto front, while the robustness of the Pareto designs varies considerably. In fact, robustness has little to do with optimality, but more with topology. Scale-free arrays like random and VLA-like are more robust compared to uniform degree arrays like circular, triangular and Reuleaux triangular.

The tension between Pareto optimality and robustness can probably be best exemplified with overlaying the best design paths based on cost and performance and the best robustness paths. Figure 4-8 shows three subsequent stages for both backward and forward with the robustness path marked with red diamonds. For both strategies, the more robust path is far behind the Pareto front and the best cost-performance trade-off path. The good news is that for different strategies, the robust design choice requires a sacrifice in only one metric. In the case of backward, robustness is achieved at greater cost but relatively the same performance. For forward, the cost to design robustly is the same as the optimal path's, but the performance is worse with a higher *uv* density. With this insight, we can clearly term the backward strategy a *cost-optimizer*, and the forward a *performance-optimizer*.

4.5 Network Robustness as a Design Objective

The next natural step after evaluating robustness of designs is to design for robustness. Substituting the energy function and the cost-performance trade-off with robustness, allows to probe which geometries are the most robust, and whether indeed, designing for robustness requires a large sacrifice in traditional engineering goals.

To illustrate the implications for the robustness metric, we look at the case of backward third stage, which is a genetic algorithm optimization routine. The histograms for a traditionally evolved

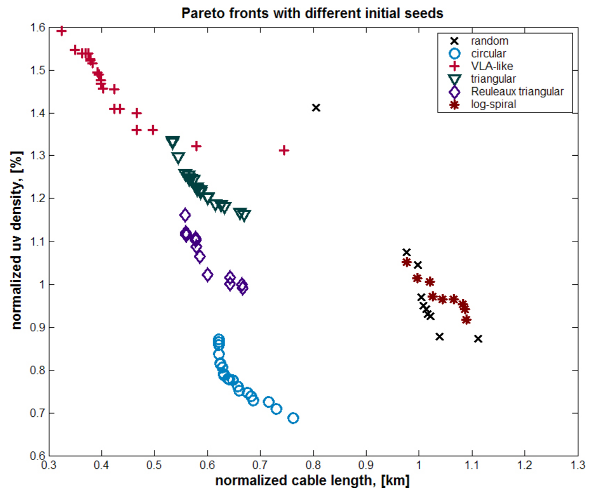


Figure 4-4: Pareto front geometries combined; second-stage backward with 60 stations; plus signs: VLA-like, diamonds: Reuleaux triangular, triangles: triangular, circles: circular, crosses: random, stars: log spirals

population and a population designed for robustness are shown on Figure 4-9 left and right respectively. The traditional population exhibits an off-centered two-peaked robustness distribution with a larger peak around 0.8. The mean for this population is 0.7584 (fraction of surviving nodes, mapping structural robustness only), minimum 0.5720, maximum 0.8870 and a standard deviation of 0.0788. The robust population on the other hand has a negatively skewed distribution with a mean of 0.8272, a maximum of 0.8940, minimum 0.6890 and standard deviation of 0.0390. The two-peaked distribution is a sign of groups of similar geometries with the same robustness characteristics and a clear indication that robustness does not arise accidentally from the design process. The skewed distribution, on the other hand, shows clear selection for robustness.

These findings can be also be seen on the color-coded plots of uv density versus cable length of different populations. The backward first stage comparison is presented in Figure 4-10. The bottom plot of the designed-for-robustness population is much lighter (more robust) than the traditional (top). Similar observations can be made in the forward second-stage picture shown on Figure 4-11. The relatively more robust design inside these populations have moved closer to the Pareto front, but more striking is the improvement of population as a whole. As the standard deviations show, the designed-for-robustness population has less contrast between individual designs.

As expected, improvement in robustness comes at a cost. The robust designs in the bottom plot on Figure 4-2 have twice the cable length. This suggests that robustness selects for different geometries as well. The next step is look at the non-dominated geometries on the uv density-cable length Pareto front.

Both backward and forward non-dominated designs are highly non-regular, characterized by many short branches allowing for short path lengths from node to node (Figure 4-12 and Figure 4-13). Decentralized architectures are rare, hubs are ubiquitous, sometimes more than one as in the last array on Figure 4-13. Multiple hub emergence is expected: many hubs, well-connected to each other are almost a prerequisite for robustness and an ingredient for hierarchical networks. If the minimal linkage requirement had not been imposed on these arrays, hierarchical (closed) formations would have emerged while optimizing for robustness.

The ten most robust configurations, backward and forward, show the same trends, star-like patterns with a lot of protruding branches. The only difference between backward and forward in the 10 most robust designs is that the forward initial arrays (in blue) are more spread out, like rugged

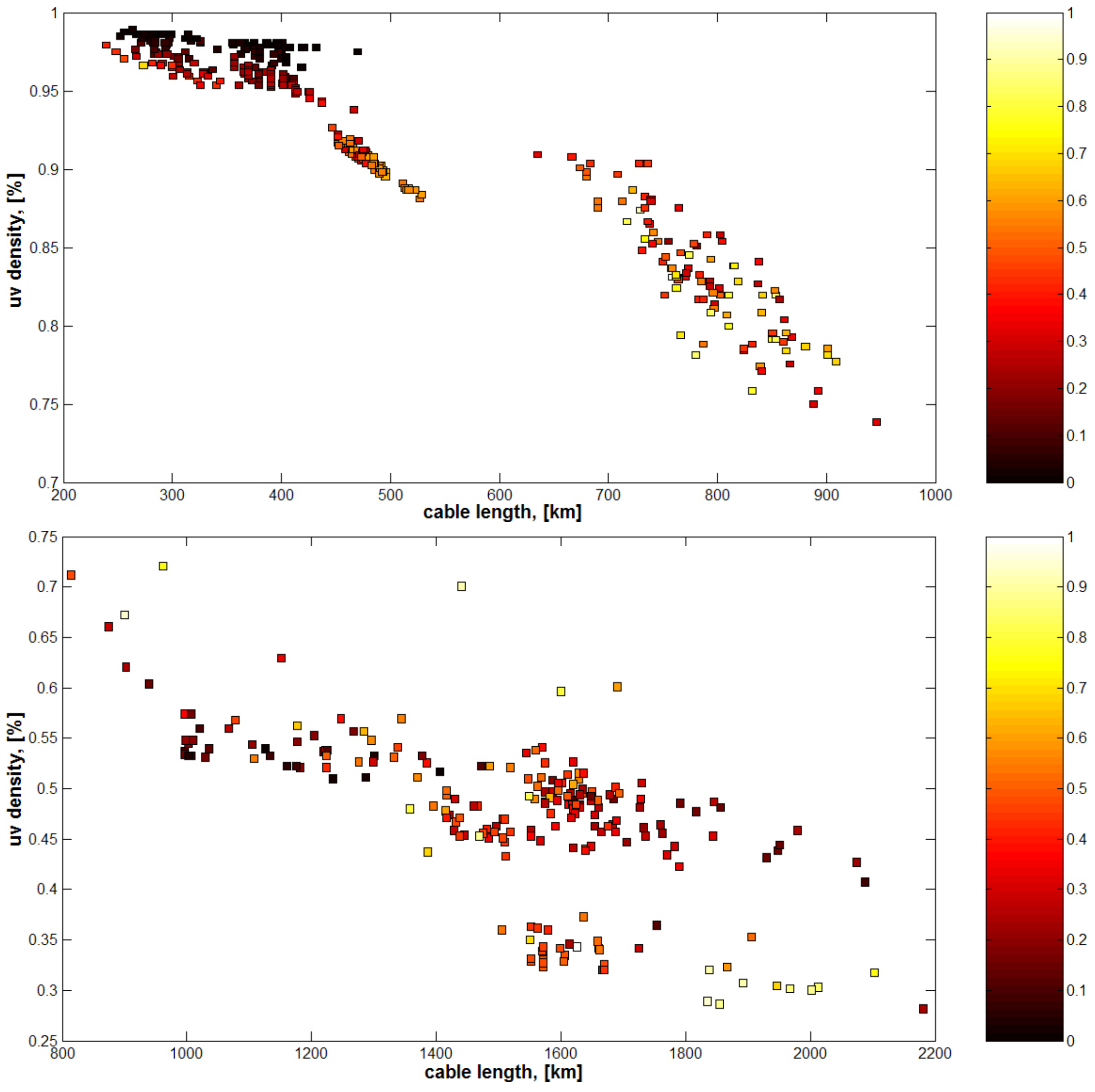


Figure 4-5: Dynamical robustness metric for first-stage backward (top) and forward (bottom), all geometries, 27 stations; robustness is indicated by the color scale, lighter is more robust

stars, with single-degree second-stage additions. The backward strategy arrays look smoother, with less curves and more tendency to form secondary hubs. The fact that the most robust designs are not distinct for the backward and the forward strategy, unlike the traditionally designed arrays, means that robustness is a characteristic which transcends the optimization process, and is entirely driven by topology.

In summary, evaluating designs for robustness revealed the role of different geometries in robust design and explained their properties with previously developed models of robust networks. Designing for robustness confirmed the geometrical results and verified that implementing robustness in the design is indeed costly, but can be worth in the long run.

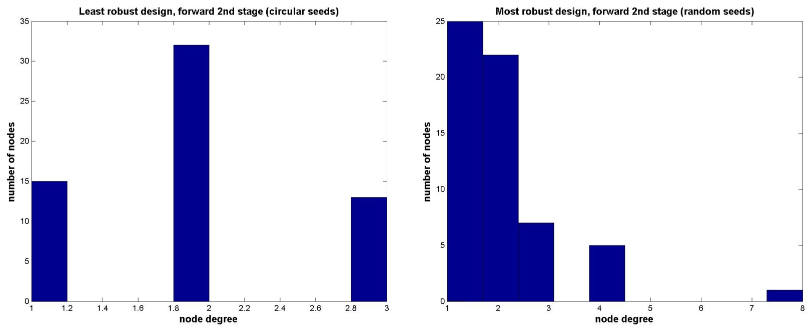


Figure 4-6: Degree distributions for the least robust (left) and most robust (right) arrays with 60 stations, forward strategy

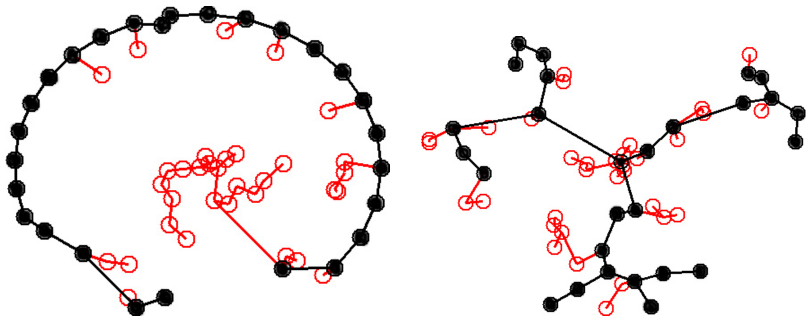


Figure 4-7: Geometries for the least robust (left) and most robust (right) arrays with 60 stations, forward strategy; filled circles indicate the original array, empty circles show the second-stage expansions

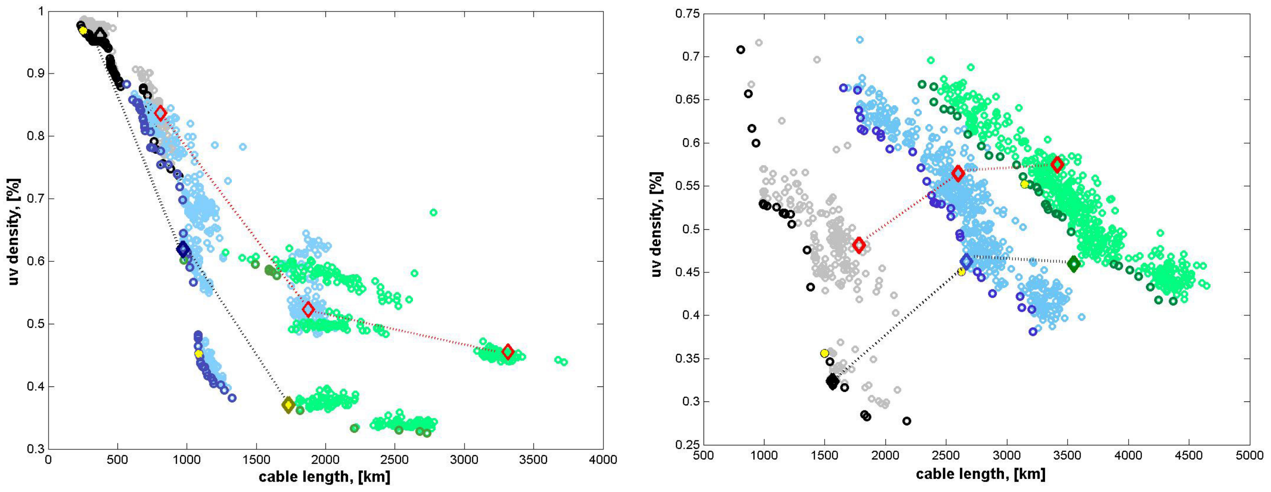


Figure 4-8: Three stages, backward (left) and forward (right), with best design paths in terms of traditional metrics (black diamonds close to the Pareto front) and robustness (red diamonds behind the Pareto front); first stage (27 stations) is shown in grey, second stage (60 stations) in blue and third stage (99 stations) in green

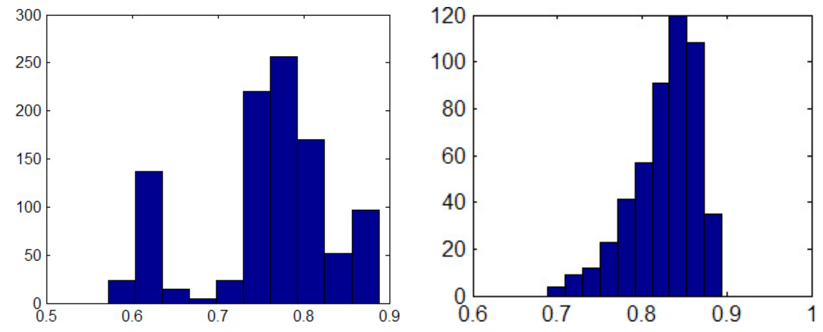


Figure 4-9: Robustness metric histograms a traditionally optimized population (left) and a population optimized for robustness (right); 24 stations

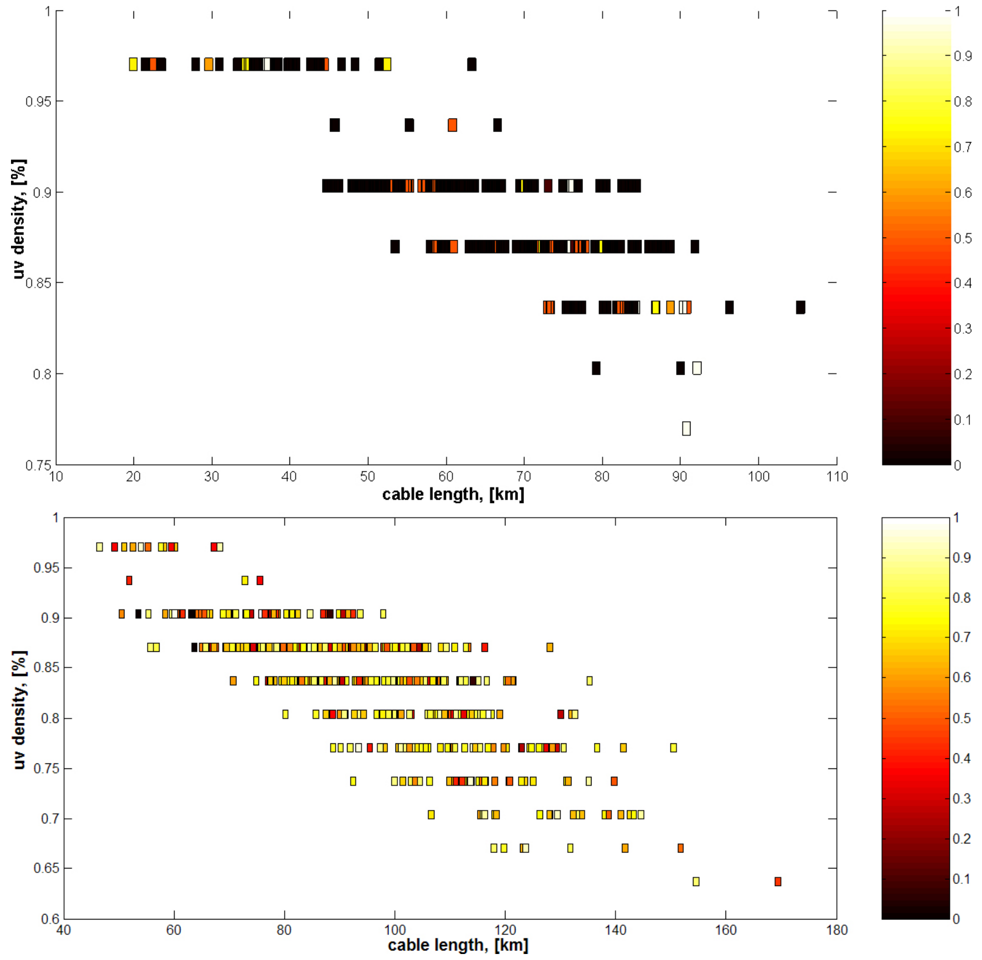


Figure 4-10: Robustness metric compared for a traditionally optimized population (top) and a population designed for robustness (bottom); first-stage backward, 6 stations

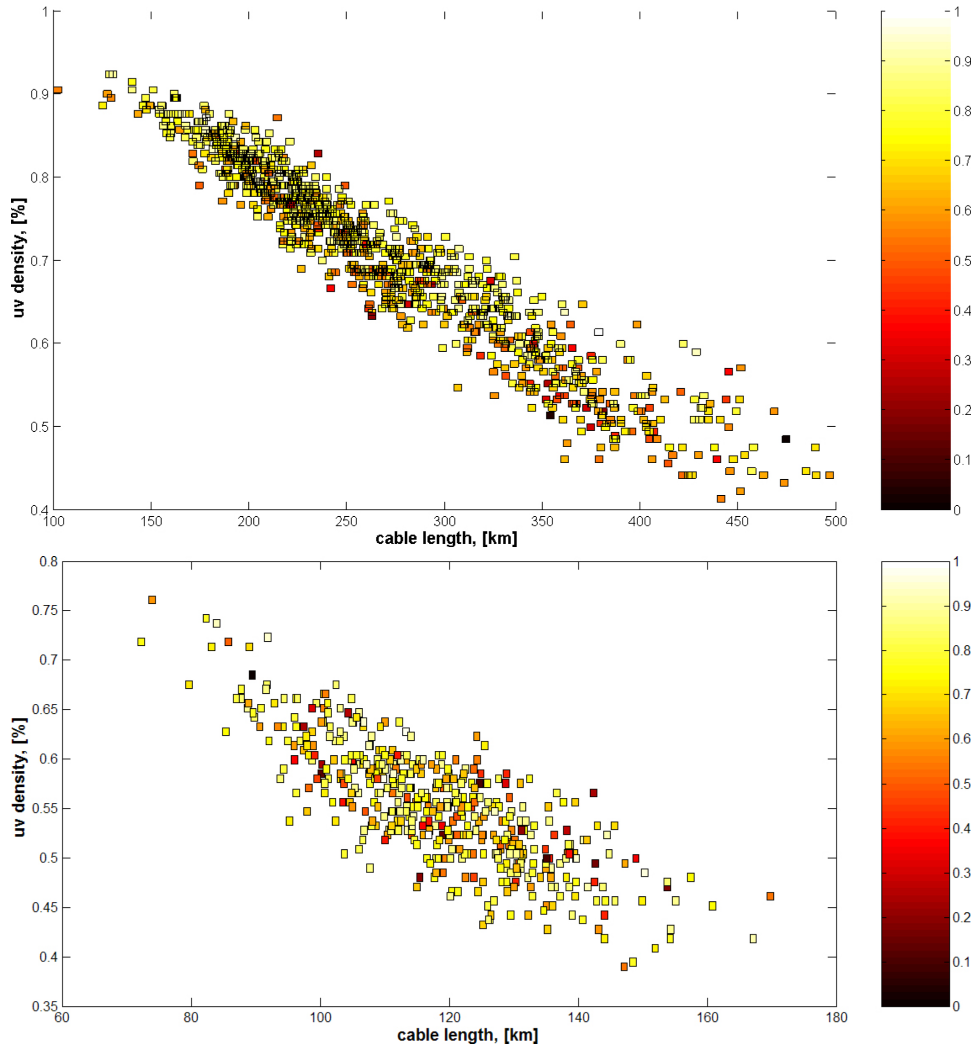


Figure 4-11: Robustness metric compared for a traditionally optimized population (top) and a population designed for robustness (bottom); second-stage forward, 15 stations

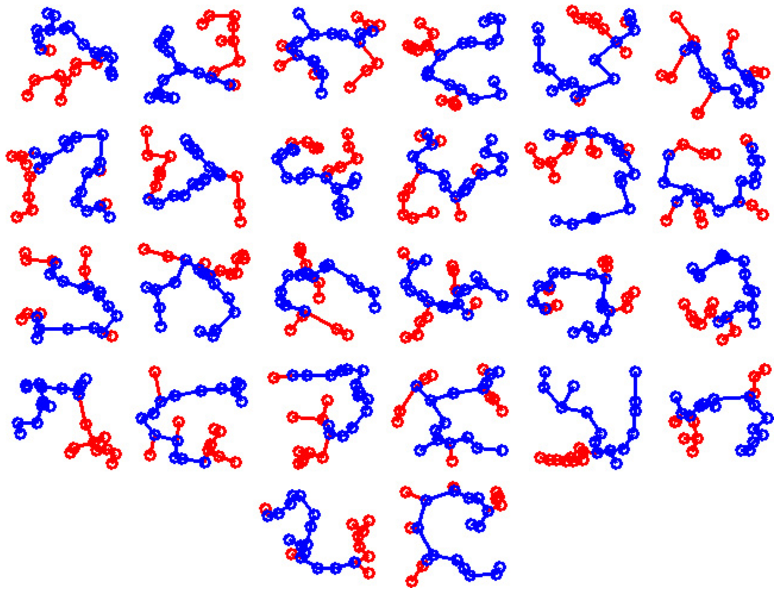


Figure 4-12: Non-dominated designs from designed-for-robustness Pareto front of *uv* density versus cable length; Pareto front based on 2^{nd} and 3^{rd} stages for backward technique with 24 stations

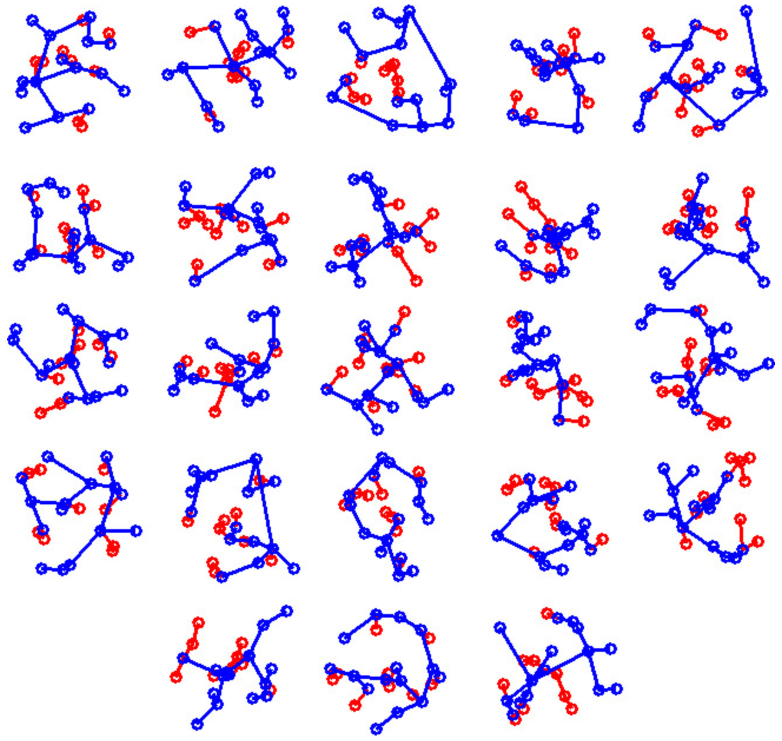


Figure 4-13: Non-dominated designs from designed-for-robustness Pareto front of *uv* density versus cable length; Pareto front based on 2^{nd} and 3^{rd} stages for forward technique with 24 stations

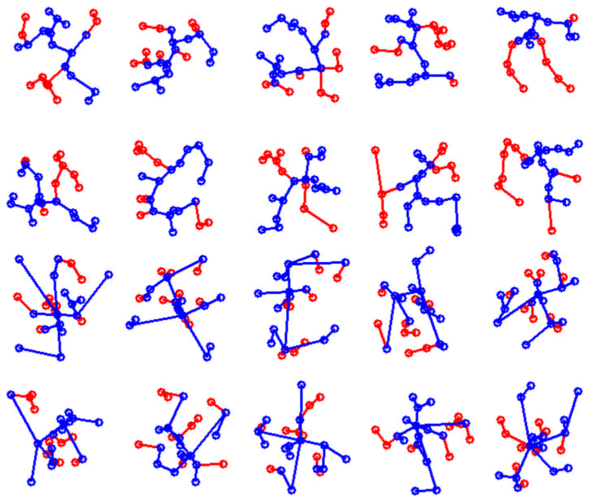


Figure 4-14: 10 most robust designs, backward (top 10) and forward (bottom 10)

Chapter 5

Discussion and Conclusions

5.1 Graph-theoretical Considerations

The purpose of the graph-theoretical considerations presented in this thesis is to both understand the results of the optimization and also to drive the design process. Measures like degree distributions, robustness histograms and average minimum path length are used to understand the difference between the topologies of robust and non-robust arrays and how their structure arises from the optimization. Using graph theory, we explained why non-Pareto designs are more robust, and what network model categories do different seed geometries fall in. For example, circular configurations are an instance of 2-regular graphs with uniform degree distribution, long average minimum path length, and as such, not among the most robust. Random geometries, VLA-like and log spiral geometries are relatively more robust because of their potential to evolve into hub-centralized configurations, even multi-hub formations, as seen in the experiments with design for robustness. Moreover, the correlations confirmed between robustness properties and graph theoretical measures like average minimum path length, suggest that the two problem-specific metrics, uv density and cable length can be replaced with more generic metrics which span the same tension between graph linkage and coverage. Our preliminary results show that using clustering coefficient as a substitute for cable length and average path length as a substitute for uv density produces a similar Pareto front.

We envision using an abstracted framework as described to model various networks along with their problem-specific objectives to gain more intuition about overarching network scaling laws and to compare the growth and evolution of seemingly unrelated problems.

5.2 Open issues

The work presented in this thesis has unveiled more questions than it has answered, revealing a rich potential for refining the underlying case study both in theoretical and experimental aspects. Moreover, the developed framework is widely applicable to a variety of domains some of which were intended to find place in this thesis originally. Unfortunately, the rich breadth and depth of these applications did not allow them to be presented here, but instead helped set the underlaying framework for further research. The open issues left to be addressed further include what can be done with more computational resources and refined algorithms, what study cases arise from challenging the problem formulation and assumptions and finally, what was left out or studied partially.

5.2.1 Computational resources

The tradeoff between computational resources and fidelity is often a limiting factor in optimization. The results' fidelity in this work has rarely suffered from insufficient time to converge. Given more time though, more simulations with more iterations, especially in the for the robustness-optimized arrays might confirm the established patterns. Robustness, for example, is one of the metrics taking the longest to compute with time growing as the square of number of nodes. It is expected that with more stringent convergence conditions, more insight can be gained as to what geometries satisfy better coverage and cable length objectives versus robustness.

5.2.2 Challenging the assumptions

Several assumptions have been made both in the physical and the simulation modeling of large telescope arrays. Challenging or revising these assumptions opens a potential for more refined studies that will provide answers to the questions raised by our results.

- Due to the continuous nature of the stations coordinates and the large design space with respect to station size, gridding the design space was avoided as requiring too much memory. Despite the cost, a gridded design space, as opposed to random coordinates generation, can provide more fidelity. Controlling the spacing and the number of possibilities with a grid for the forward strategy would allow explicit control of all possibilities and rigorous comparison between the backward and forward strategies. Gridding also means better control of the distance between stations which would avoid degenerate clustered designs (highly optimized for low cable length). Certainly, large grids might provide less fidelity than even the random coordinates generation, so we recommend grid size on the order of a single station diameter.

- The minimum separation distance between stations assumed for the new stations generation is 1 kilometer. Allowing a different minimum distance might result in different topologies for the forward strategy, though in general we expect more costly, better coverage designs.

- The minimal spanning tree algorithm for optimum linkage deserves to be challenged, primarily for robustness purposes. Allowing multiple paths from node to node will definitely make the array more robust to both node and edge failure. It is worth performing a study of coupled cost, performance and robustness of Steiner graphs versus arrays with less optimal linkage.

- The design vector perturbation only assumed one-variable substitutions. It has been suggested that three-variable substitutions are the optimum for swift exploration of the design space. Challenging this assumption in the DOE algorithm might allow the exploration of larger parts of the design space and achieve faster convergence.

- As repeatedly mentioned, the poor performance of the backward in the first stage is due to the stringent connectivity constraint. Allowing any subset of the original array in the backward strategy to be a valid option will certainly improve performance because it will allow spread out stations to form arrays, even though they might be initially connected. One way of allowing more spread out subsets to be valid is to allow non-neighboring stations that are still connected via cable (going through another stations) to be neighbors in the optimal subset array. By any means, it is worth studying the trade-off of additional *recabling* cost versus the poor performance obtained by keeping the initial infrastructure. In some way, challenging the connectivity assumption makes the backward strategy more *forward-like.*

- In the search for a global optimum, the forward search can be improved by generating more random stations for each array and thus increasing the space of possibilities. A full DOE would involve gridding the design space and searching every grid possibility.

- In our work, the telescope stations are assumed to be points in two-dimensional space, not physical apertures. A more refined study can model the physics better by creating a continuous model for every telescope that updates at discrete intervals or communicates with other nodes in the network at discrete intervals. Such a continuous-discrete network model provides more fidelity in the modeling and allows continuous feedback in monitoring performance throughout the design process. We expect that physical modeling will uncover engineering problems that need to be addressed in the network context.

5.2.3 The remaining cases

Apart from starting with new assumptions, there are remaining cases which are allowable or extendable from the existing framework, but which were not explored too keep the consistency and simplicity of the original model. Those might be of value in further studies.

- At present, in a single stage stations are meant to operate at the locations designed. One might conceive of designing regular fixed paths along which stations could move to fine-tune the performance based on the observation goal at hand. This is not a new idea, implemented to some extent by the VLA array in New Mexico [23]. Reconfiguration onsite will drive the costs of positioning, motion and cabling aspects higher, but it will provide great flexibility even in a single-stage design. An interesting question is what kind of geometries favor a single-stage reconfigurable design, both from a cost and performance perspective.
- In most networks, not all nodes are equal. A hub is a small functional differentiation, but networks can have structurally hybrid nodes, modeled with different sizes (larger and smaller clusters of telescopes), cost and functionality. Links might also vary in importance, cost, speed, reliability and so on.
- Experiments can be performed with varying objective weights in the energy function. It is expected that this will result in more refined network topologies and a more complete Pareto front. Partial work on varying weights has already started.
- Stages can be weighted other than equal, by giving priority to the future or present design. This will change the vision for the backward and forward strategies and maybe require a new strategy.
- The number of stations has been kept constant at every stage. A valid question is to ask, what is the optimal number of stations given a certain budget and a desired performance. We have partially studied this problem with a feedback algorithm that initially guesses a reasonable number of stations and then re-designs the array based on a budget limit. The resources left from downsizing are then assigned to the next stage and the procedure is repeated. A problem formulation which allows the number of stations to vary opens many more questions, like what geometries grow best under cost constraints or are more flexible to downsizing. Answers to these questions could lead to designing more adaptable systems.
- Simulated annealing might not be the only algorithm suitable for network growth optimization. While a genetic algorithm might be too computationally extensive, research into other possibilities will be useful to benchmark the SA results.

5.3 Potential Application to Other Domains

We believe that the framework developed for telescope array optimization is general enough to be employed in various other problems, susceptible to network representation. First, our models can be mapped to the growth of existing systems, tested, validated, updated and augmented based on this knowledge. An example is the growth of cellular coverage networks, which have naturally extended from large cities, to large commuter arteries and then further on to small towns and less inhabited places. The expanding map of Internet routers and hyperlinks is also well-documented and an example of demand-driven *natural* growing system. Some transportation networks, like the Tokyo subway system, have also grown spontaneously, without planning, based on where need has arisen. In the case of Tokyo, as different parts of the city developed as commercial, cultural and financial centers, subway lines were built to service this need. Quite unfortunately, this city also provides a good model for random and deliberate attacks due to frequent earthquakes and the bombings at the beginning of the nineteenth century. The subway, as well as other transportation networks, has been frequently rebuilt, extended or repaired where needed. Studying the robustness of such a real network can be very exciting and revealing.

Design problems, like the studied telescope arrays, are another potential application. First of all, design recommendations can be made for or extracted from all of the natural growth studies described above. There is a myriad of potential design applications from extensible satellite networks, modularized space exploration systems, wireless sensor networks, to airline hub systems. A lot of these example problems will require hybrid nodes in their models, and extended physical models, coupled with the generalized network architecture.

5.4 Computational and Visualization Strategies

Folk wisdom has it that there are as many answers as there are questions, and as many interpretations as there are stories. While this philosophy has worked well in the past, engineers more and more recognize the need for overarching models that capture larger types of problems. It is our hope that the framework developed for the presented research, though constructed to solve the particular problem of large telescope arrays, can be generalized to a variety of network optimization problems. With this goal in mind from the start generalized algorithms like simulated annealing were adapted to different cases, like backward and forward optimization. Both genetic algorithms and simulated annealing turned out to be well-suited for network problems with discrete state transitions and large continuous design spaces.

Visualization techniques have been an indispensable analysis tool in this work. Visualizing array geometries for sizes up to 100 nodes can still help to identify regular patterns and emerging geometries. For real networks, explicit graph visualization is not always useful, but histograms of vertex degrees and other network properties give an implicit picture of the system. For example, a power-law degree distribution indicates hierarchy in the network and an abundance of low-degree nodes. Visualization has also been widely useful in understanding and designing various algorithms, like the design vector perturbation (walking on the graph) for the backward DOE routine. Moreover, most of the results in this thesis are presented in graphical format. The precise coordinates of the best designs are not as revealing as their relative position on the Pareto front or their physical geometry. Understanding the power of visualization in network theory, we have already began and plan to continue using more visualization-enabled software like *Pajek* [24].

5.5 Conclusions. Support or Refute Hypothesis

This thesis extended previous work done on network optimization of telescope arrays to multiple stages. It was confirmed that the best extensible configurations are not the single-stage winners, due to penalties for extensibility paid in the first or last stage of the design depending on the strategy. Another hypothesis confirmed is that robust arrays do not reside at the trade-off front of cost and performance, but are always non-optimal. Designing and evaluating arrays for robustness also explained the appearance of certain geometries with previously developed network models.

Perhaps the most surprising conclusion from this work is the refuted hypothesis that the backward staging strategy is superior for the staging of telescope arrays. It was concluded instead that backward is suitable only when the end state of the system is the primary goal, for which intermediate building blocks are needed. For sustained performance throughout all stages, embedded flexibility in the design to future budget or demand uncertainties, forward is the recommended strategy.

Appendix A

Figures

A.1 Pareto plots

Figures A-1 to A-10 each contain two plots, one for the backward staging case and the other for forward. The first five plots are generated with 6, 15 and 24 stations for the three stages. The second five (Figure A-6 to A-10) represent the case for three stages with 27, 60 and 99 stations respectively. There are five figures for each case because five initial seeded geometries were considered: VLA-like, circular, Reuleaux triangular, triangular (all regular geometries) and random. Each plot contains three graphs of *uv* density versus cable length, one for each stage and one combined for the two stages. Pareto fronts are presented in red, versus the entire population in black.

A.1.1 6 to 15 to 24 stations

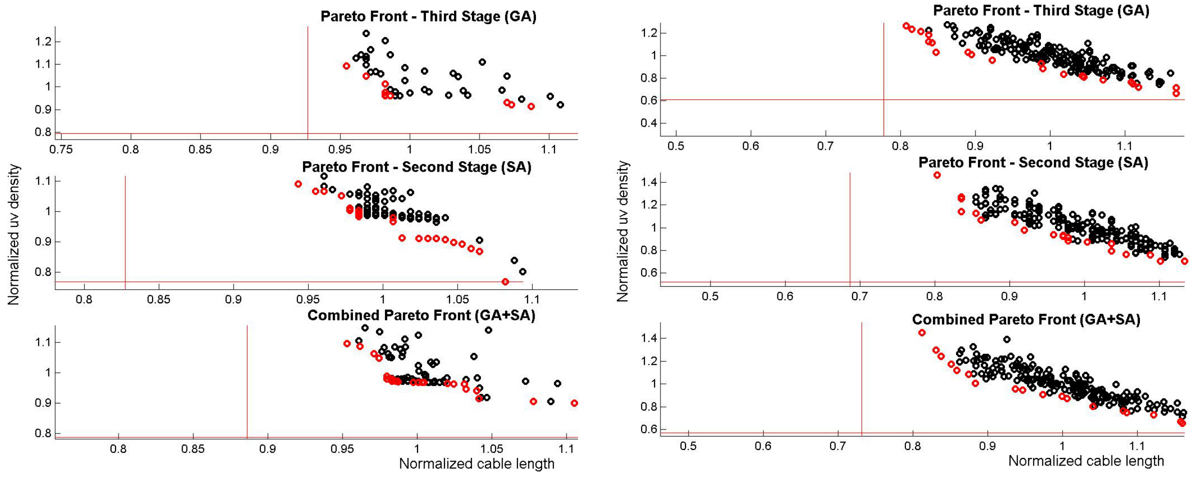


Figure A-1: Relative Pareto fronts for VLA-like seeded configurations (6, 15, 24), (a) backward (b) forward

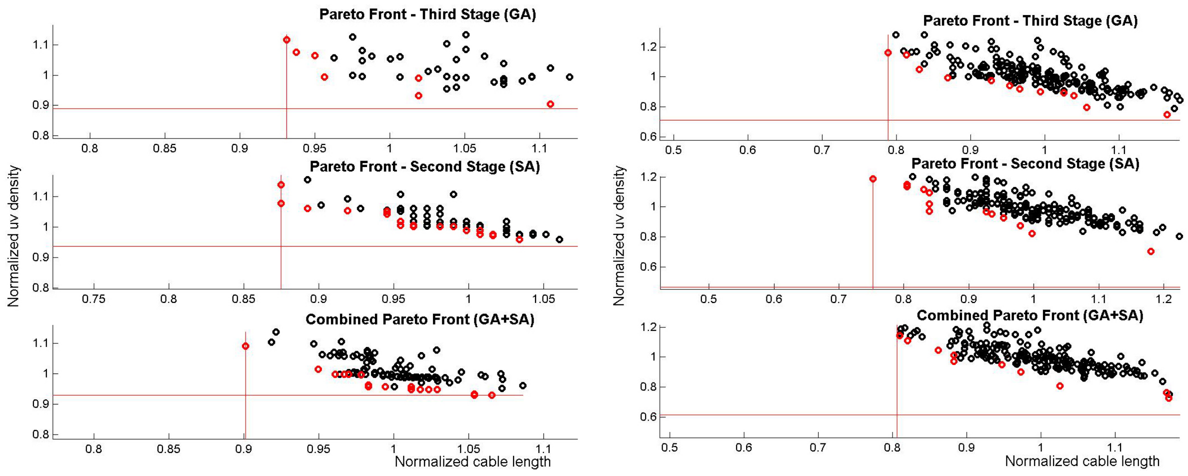


Figure A-2: Relative Pareto fronts for circular seeded configurations (6, 15, 24), (a) backward (b) forward

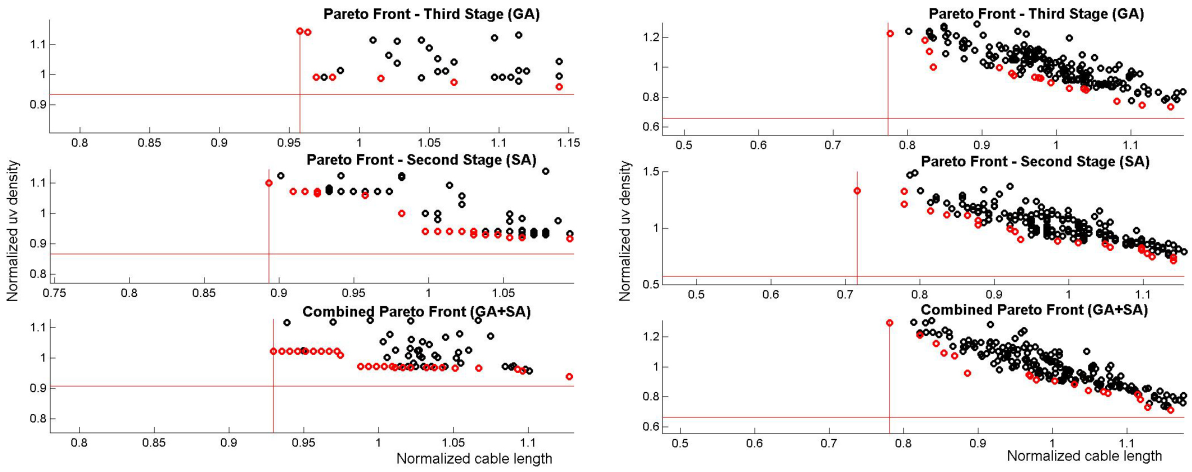


Figure A-3: Relative Pareto fronts for Reuleaux-triangular seeded configurations (6, 15, 24), (a) backward (b) forward

A.1.2 27 to 60 to 99 stations

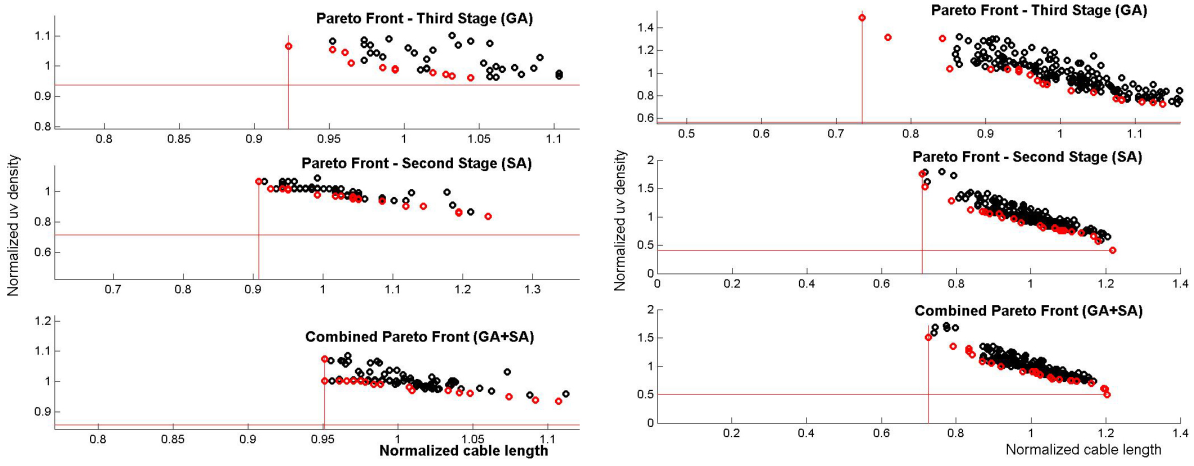


Figure A-4: Relative Pareto fronts for triangular-seeded configurations (6, 15, 24), (a) backward (b) forward

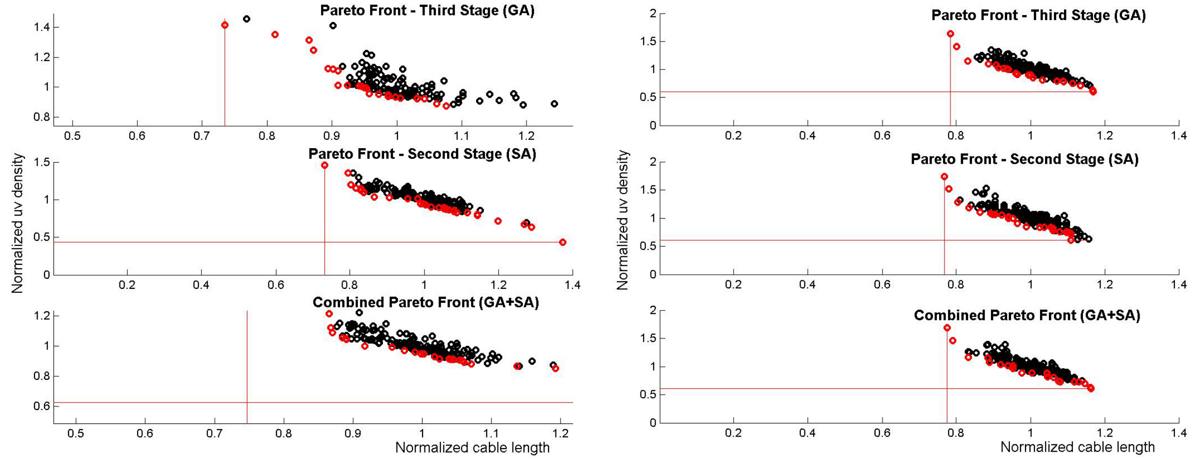


Figure A-5: Relative Pareto fronts for randomly seeded configurations (6, 15, 24), (a) backward (b) forward

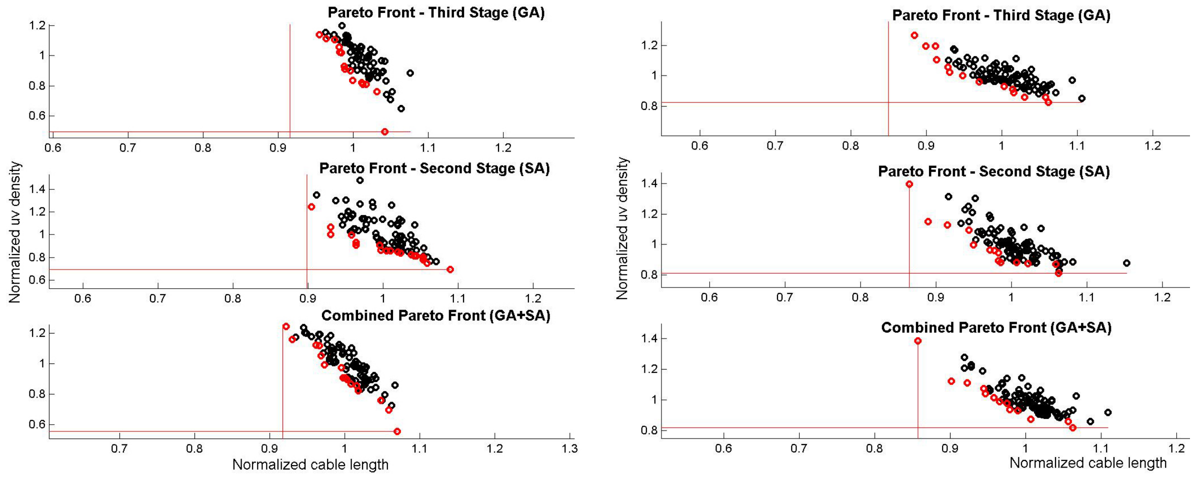


Figure A-6: Relative Pareto fronts for VLA-like seeded configurations (27, 60, 99), (a) backward (b) forward

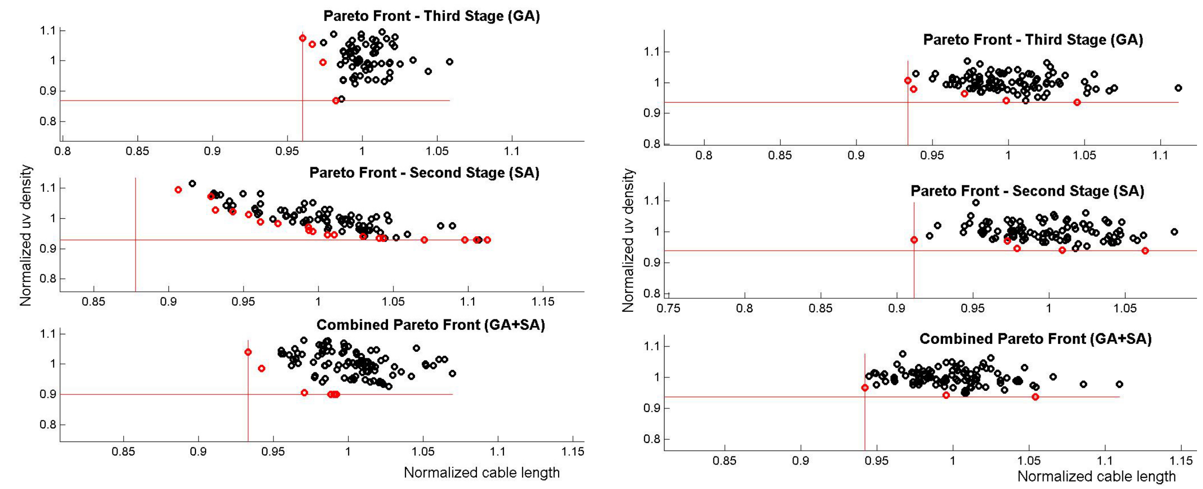


Figure A-7: Relative Pareto fronts for circular seeded configurations (27, 60, 99), (a) backward (b) forward

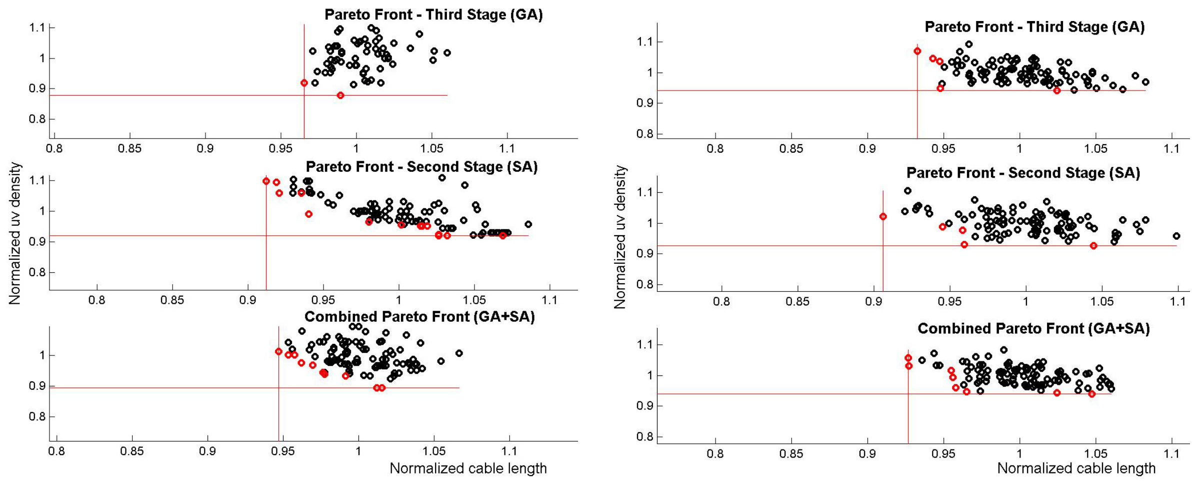


Figure A-8: Relative Pareto fronts for Reuleaux-triangular seeded configurations (27, 60, 99), (a) backward (b) forward

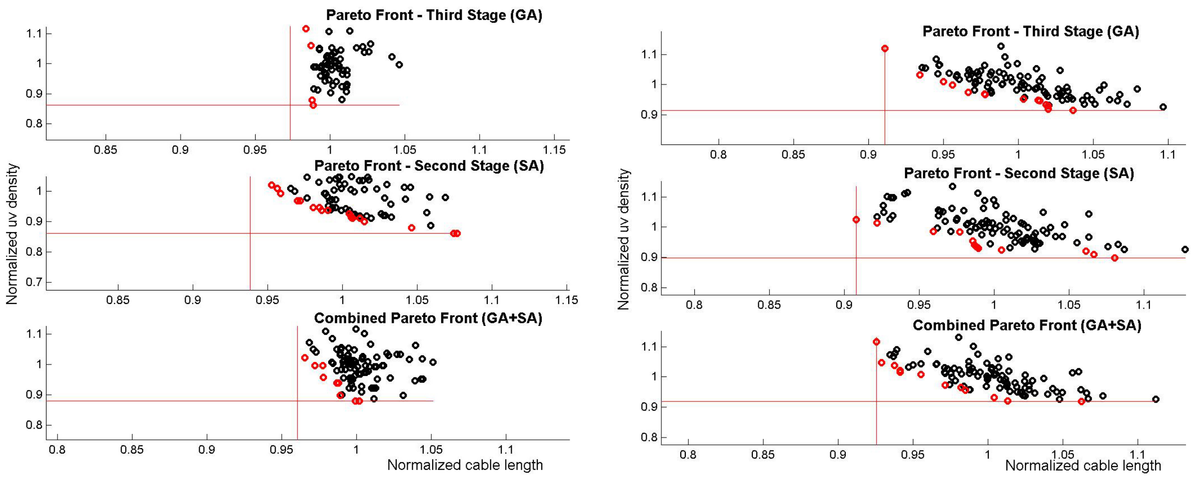


Figure A-9: Relative Pareto fronts for triangular-seeded configurations $(27, 60, 99)$, (a) backward (b) forward

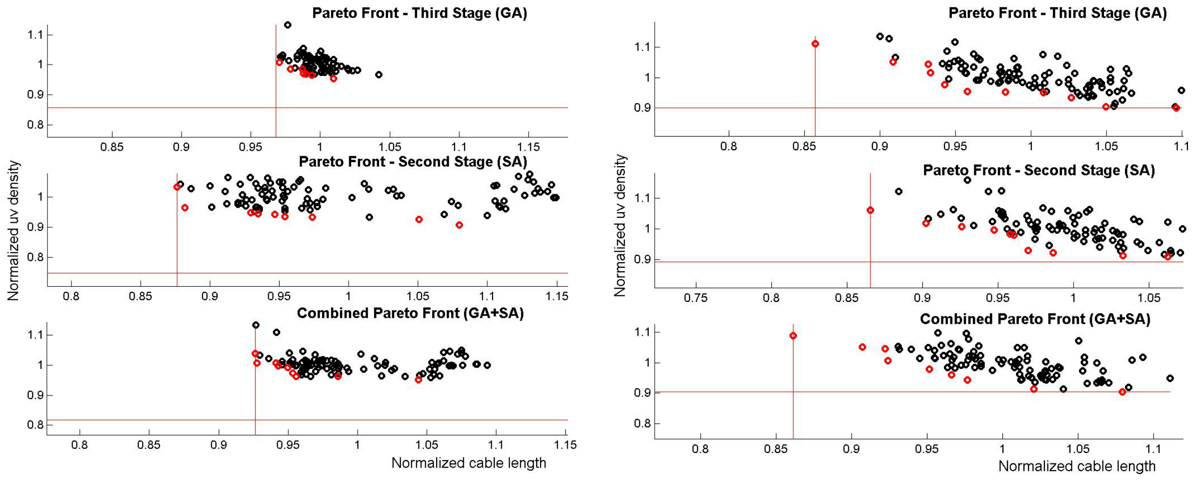


Figure A-10: Relative Pareto fronts for randomly seeded configurations $(27, 60, 99)$, (a) backward (b) forward

A.2 Convergence Plots

A.2.1 Evolving Pareto fronts $(6, 15, 24)$

Evolving Pareto fronts involves comparing Pareto fronts after optimization. All plots in this subsection show in black the initial Pareto front and in red, the evolved (converged) Pareto front. Similarly, to the previous Pareto plots, all five seeded geometries are shown for the small and larger number of stations from Figure A-11 to Figure A-20. Each figure contains four plots for the two stages, and two techniques, backward and forward (top plots are backward, left plots are first stage).

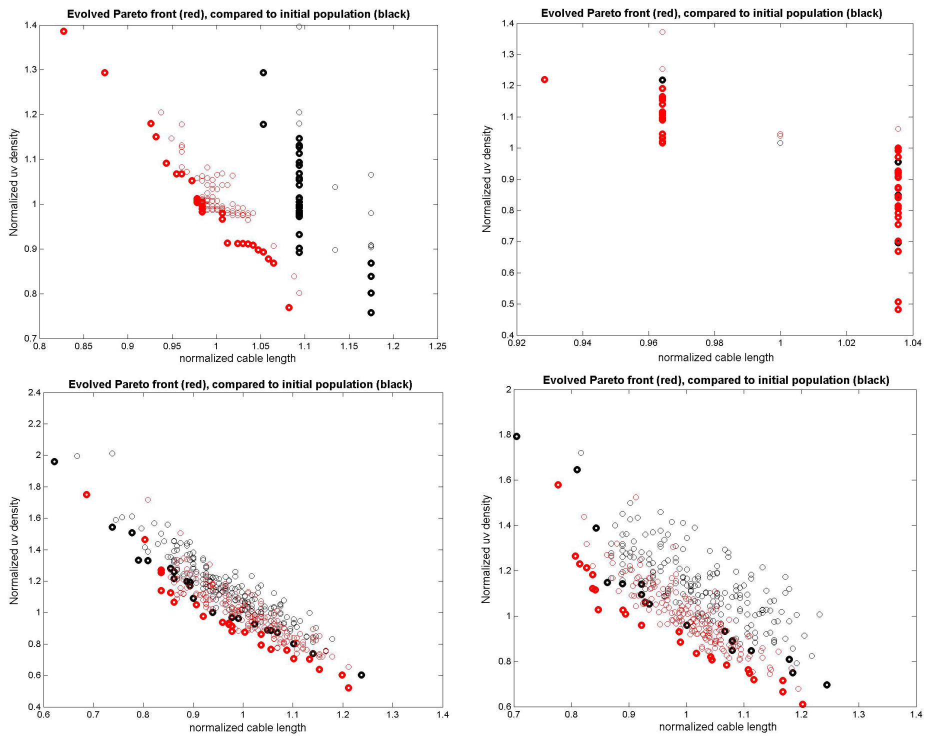


Figure A-11: Evolving Pareto fronts, VLA-like seeds; left: 2^{nd} stage, right: 3^{rd} stage; top: backward, bottom: forward path

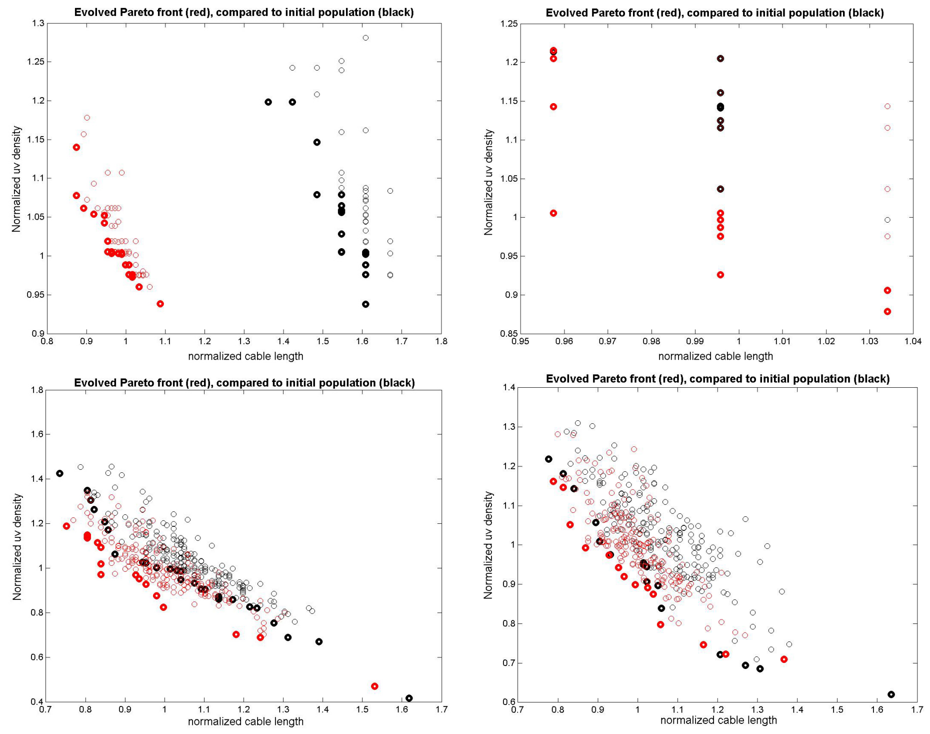


Figure A-12: Evolving Pareto fronts, circular seeds; left: 2nd stage, right: 3rd stage; top: backward, bottom: forward path

A.2.2 Evolving Pareto fronts $(27, 60, 99)$

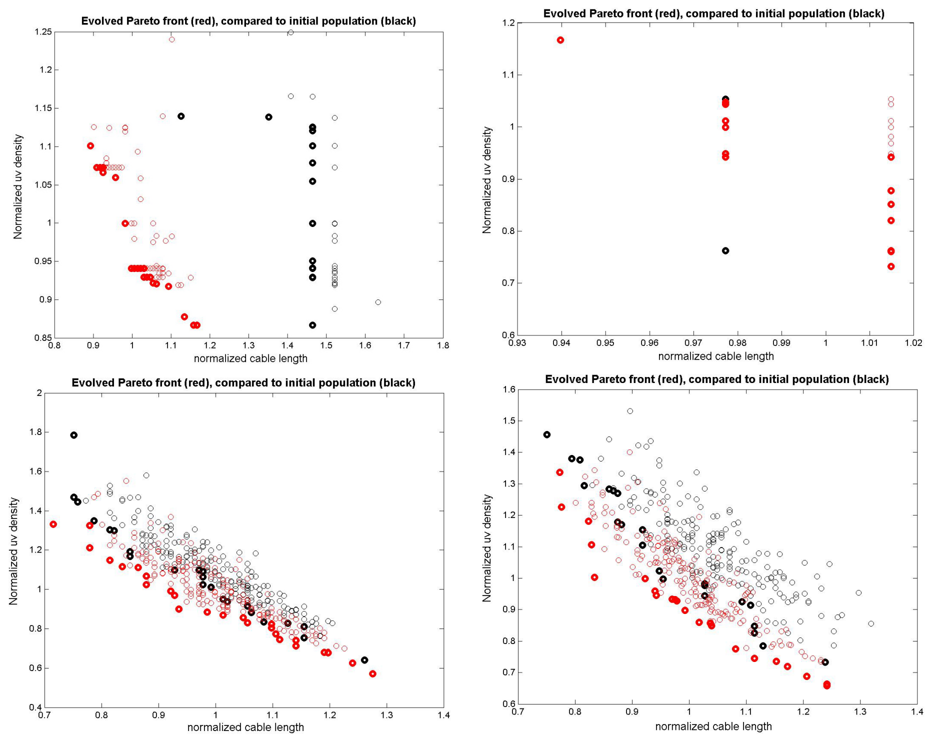


Figure A-13: Evolving Pareto fronts, Reuleaux triangular seeds; left: 2nd stage, right: 3rd stage; top: backward, bottom: forward path

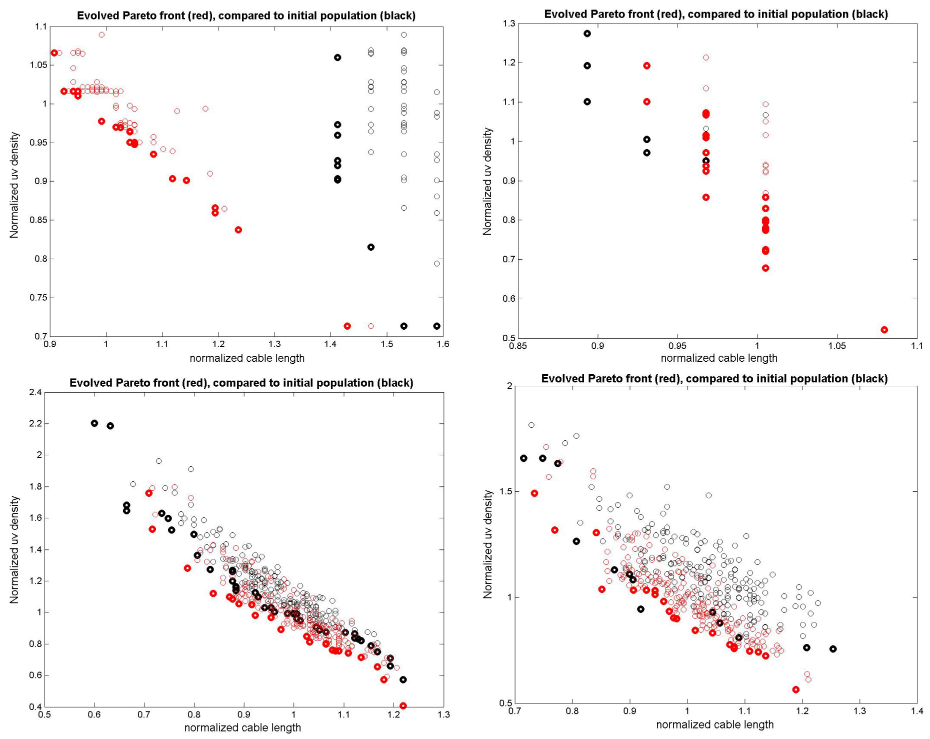


Figure A-14: Evolving Pareto fronts, triangular seeds; left: 2nd stage, right: 3rd stage; top: backward, bottom: forward path

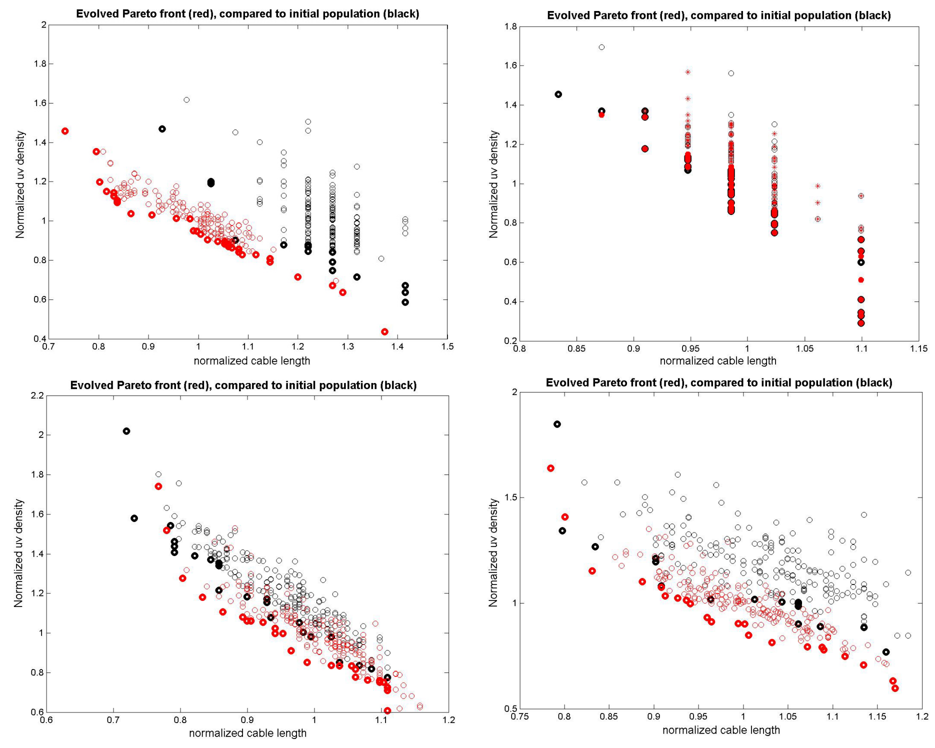


Figure A-15: Evolving Pareto fronts, random seeds; left: 2nd stage, right: 3rd stage; top: backward, bottom: forward path

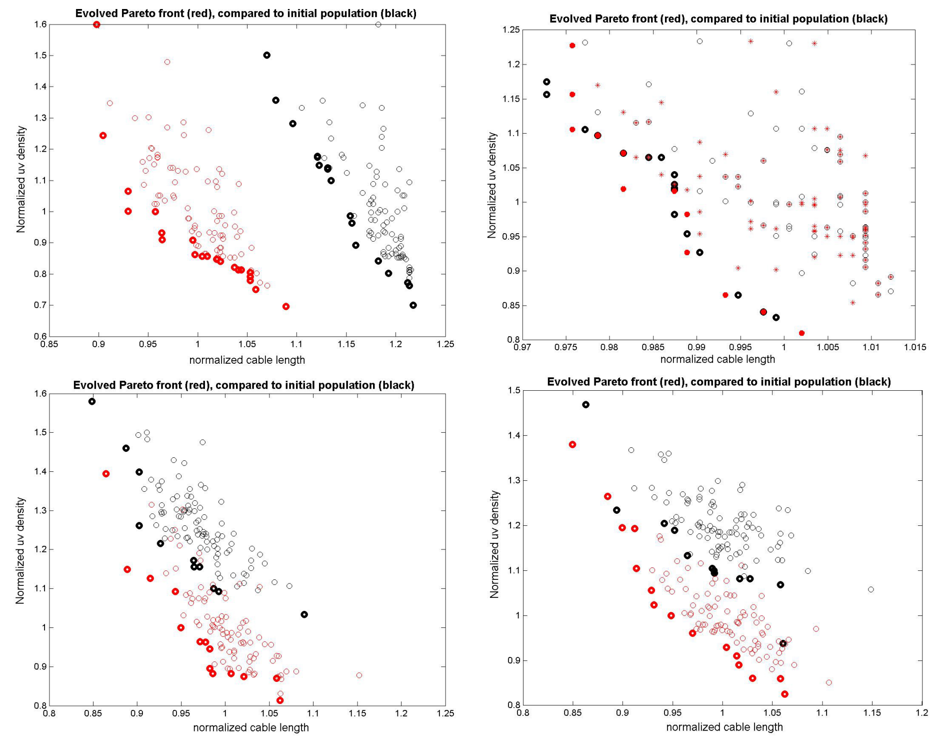


Figure A-16: Evolving Pareto fronts, VLA-like seeds; left: 2nd stage, right: 3rd stage; top: backward, bottom: forward path

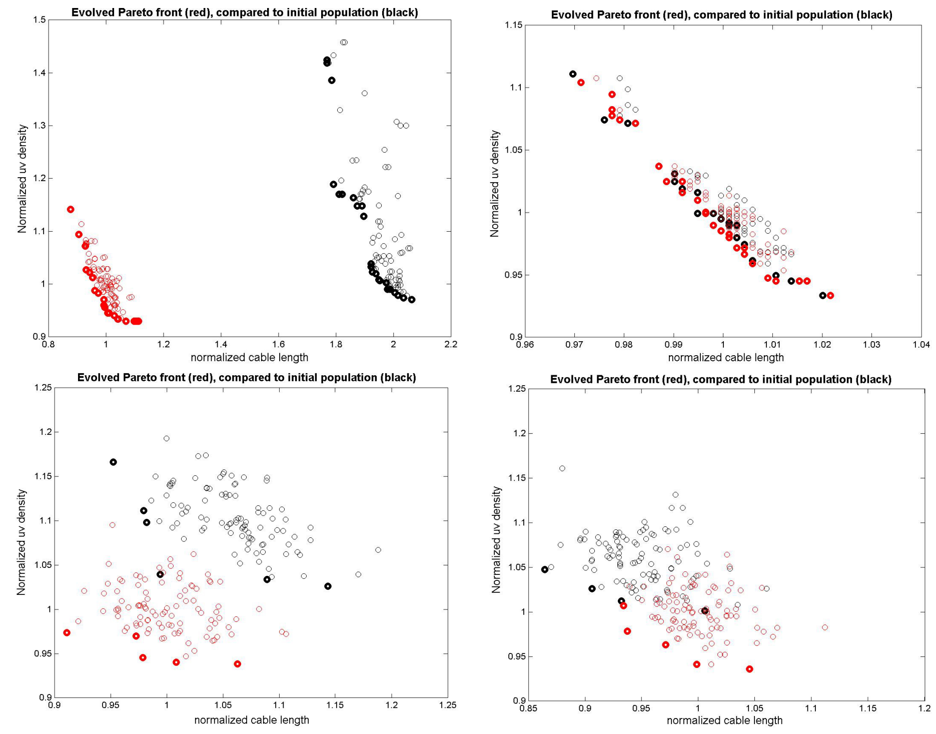


Figure A-17: Evolving Pareto fronts, circular seeds; left: 2nd stage, right: 3rd stage; top: backward, bottom: forward path

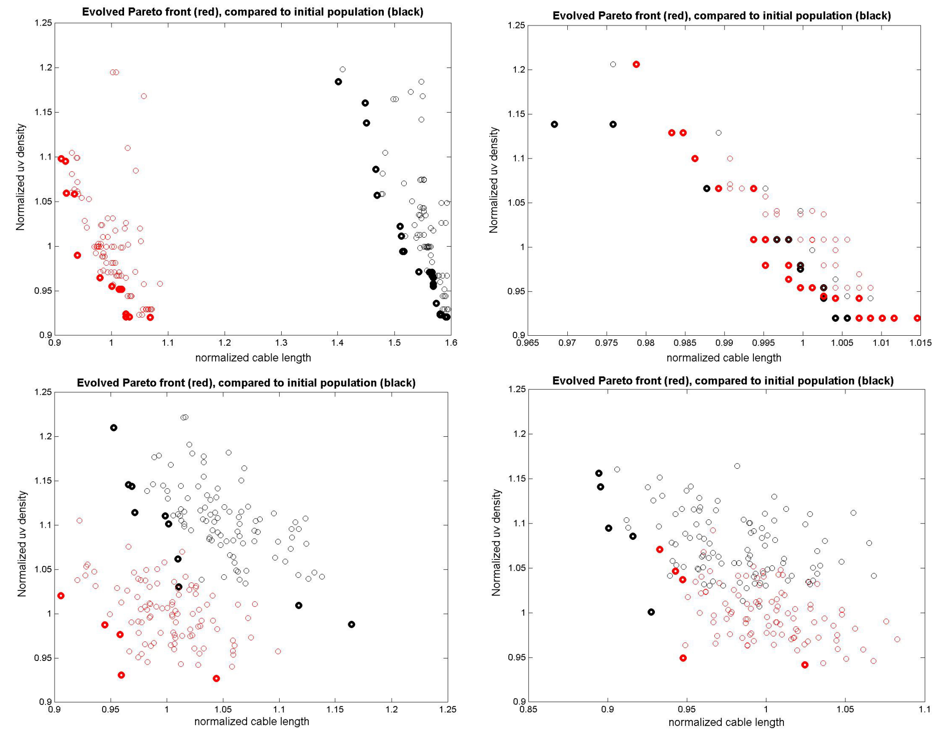


Figure A-18: Evolving Pareto fronts, Reuleaux triangular seeds; left: 2nd stage, right: 3rd stage; top: backward, bottom: forward path

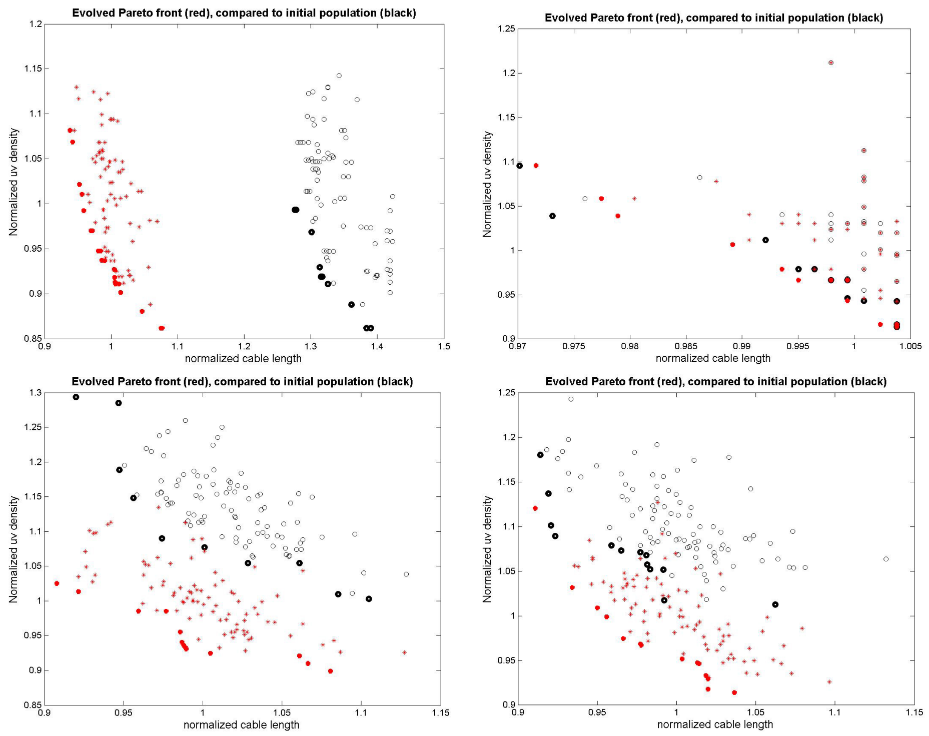


Figure A-19: Evolving Pareto fronts, triangular seeds; left: 2nd stage, right: 3rd stage; top: backward, bottom: forward path

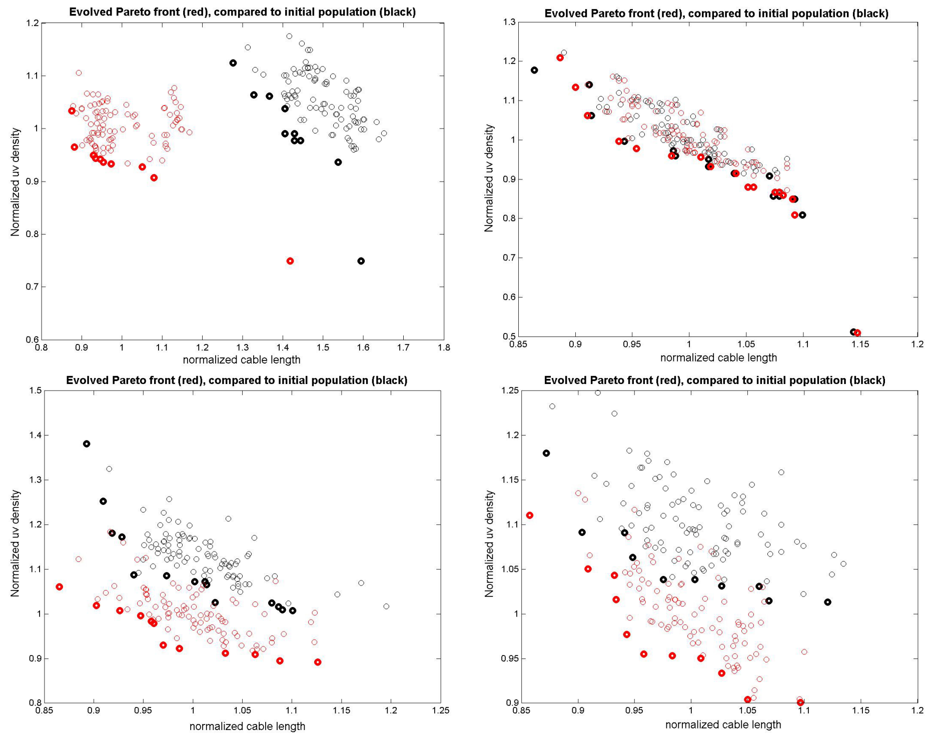


Figure A-20: Evolving Pareto fronts, random seeds; left: 2nd stage, right: 3rd stage; top: backward, bottom: forward path

A.2.3 Nadir-point convergence history

All of the convergence history plots below contain three graphs: convergence history of the energy (weighted combination of metrics), and of the two metrics, *uv* density and cable length. The five seeded geometries, (VLA-like, circular, Reuleaux triangular, triangular and random), four plots are included in each figure. The top plots show the small-number-of-stations results, the bottom show the large-number-of-stations results. Left-most plots are generated using backward staging, right-most forward staging.

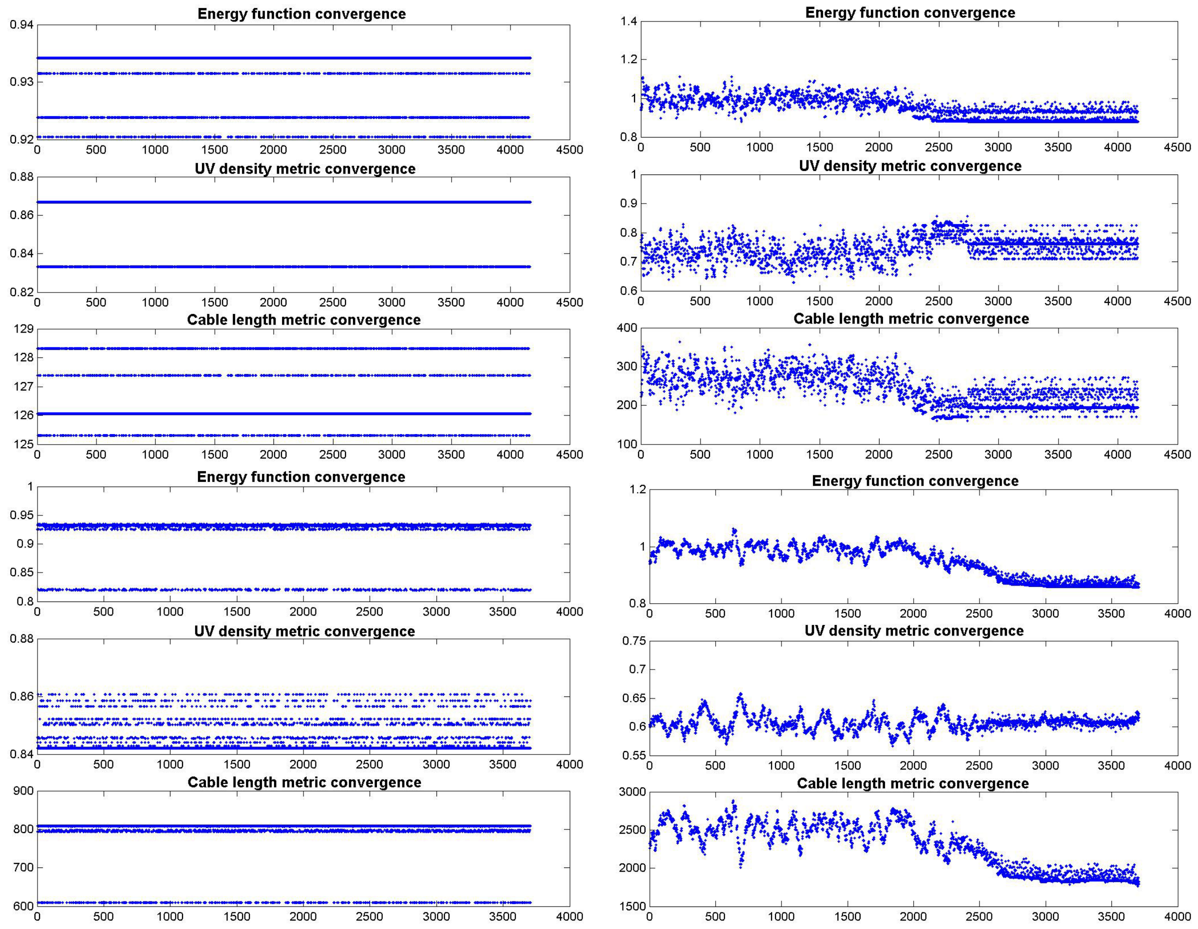


Figure A-21: Energy, uv density and cable length convergence histories; top: $(6, 15, 24)$ stations, bottom $(27, 60, 99)$ stations; left: backward path, right: forward path; VLA-like seeds

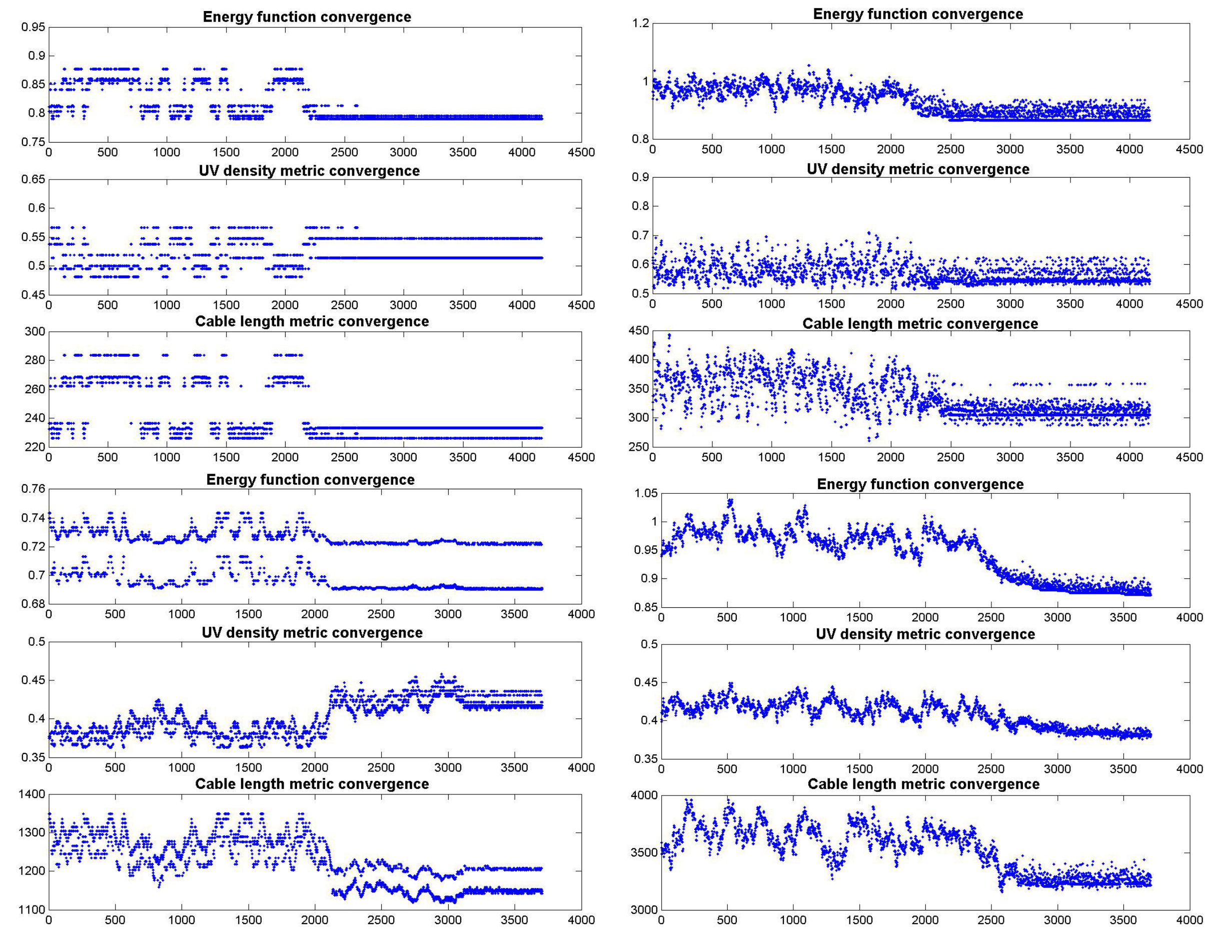


Figure A-22: Energy, uv density and cable length convergence histories; top: $(6, 15, 24)$ stations, bottom $(27, 60, 99)$ stations; left: backward path, right: forward path; circular-like seeds

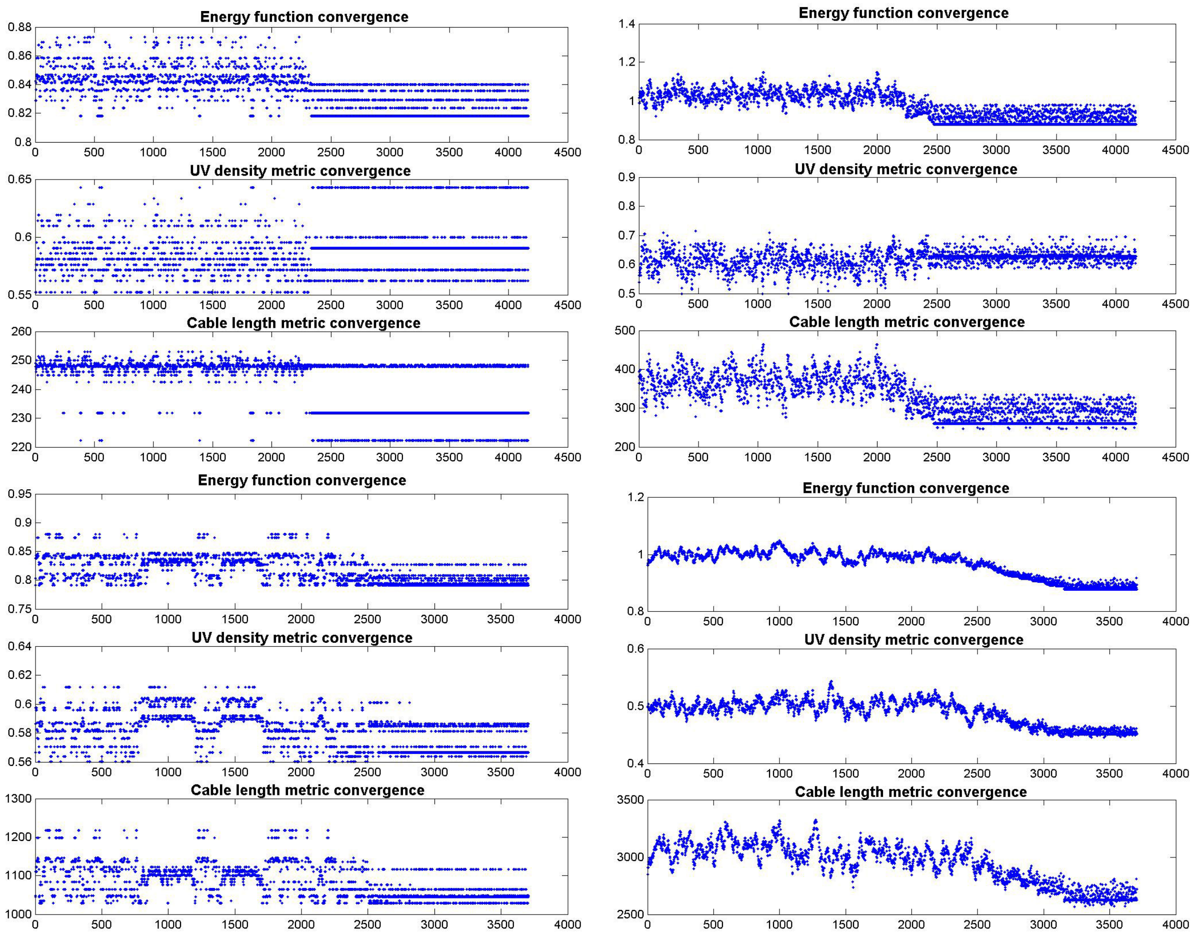


Figure A-23: Energy, uv density and cable length convergence histories; top: $(6, 15, 24)$ stations, bottom $(27, 60, 99)$ stations; left: backward path, right: forward path; Reuleaux triangular seeds

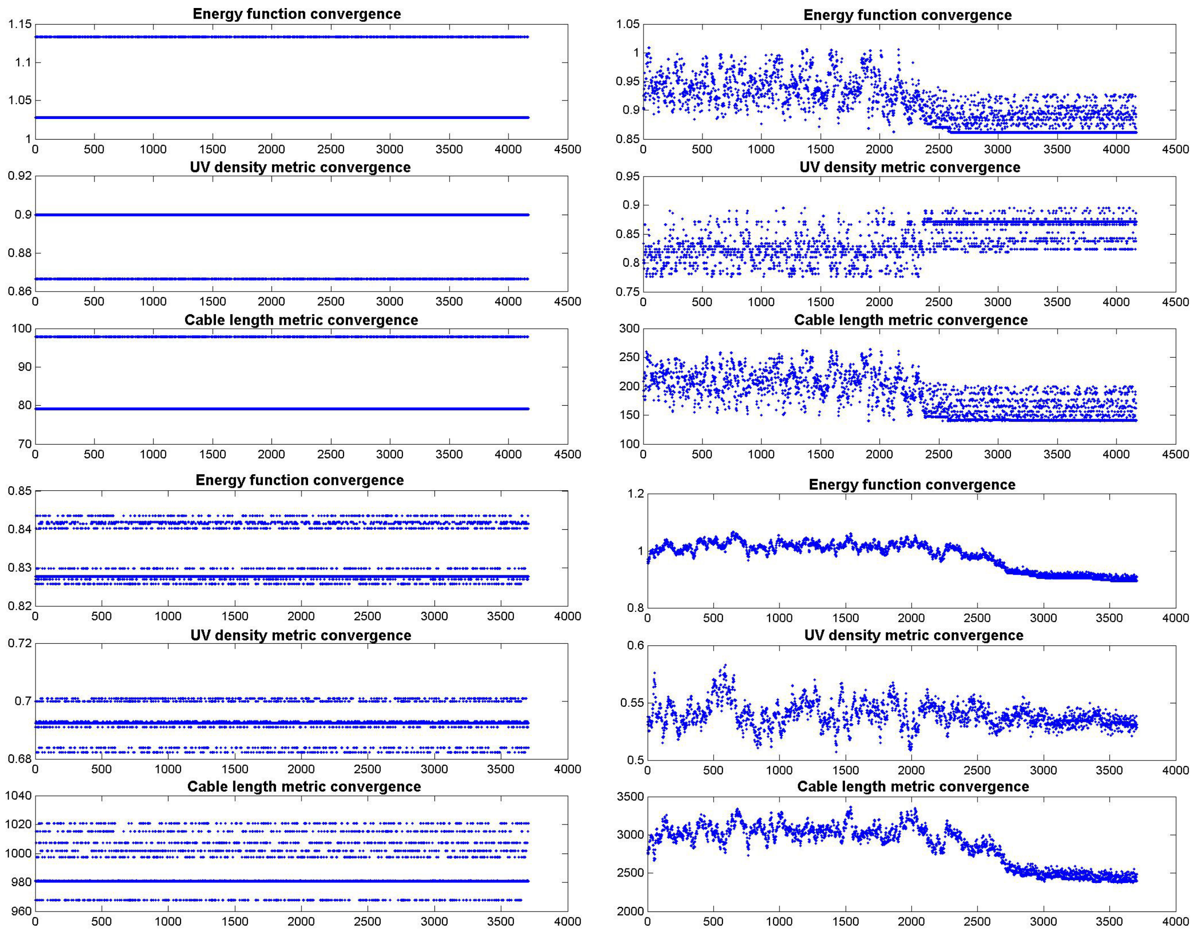


Figure A-24: Energy, uv density and cable length convergence histories; top: $(6, 15, 24)$ stations, bottom $(27, 60, 99)$ stations; left: backward path, right: forward path; triangular seeds

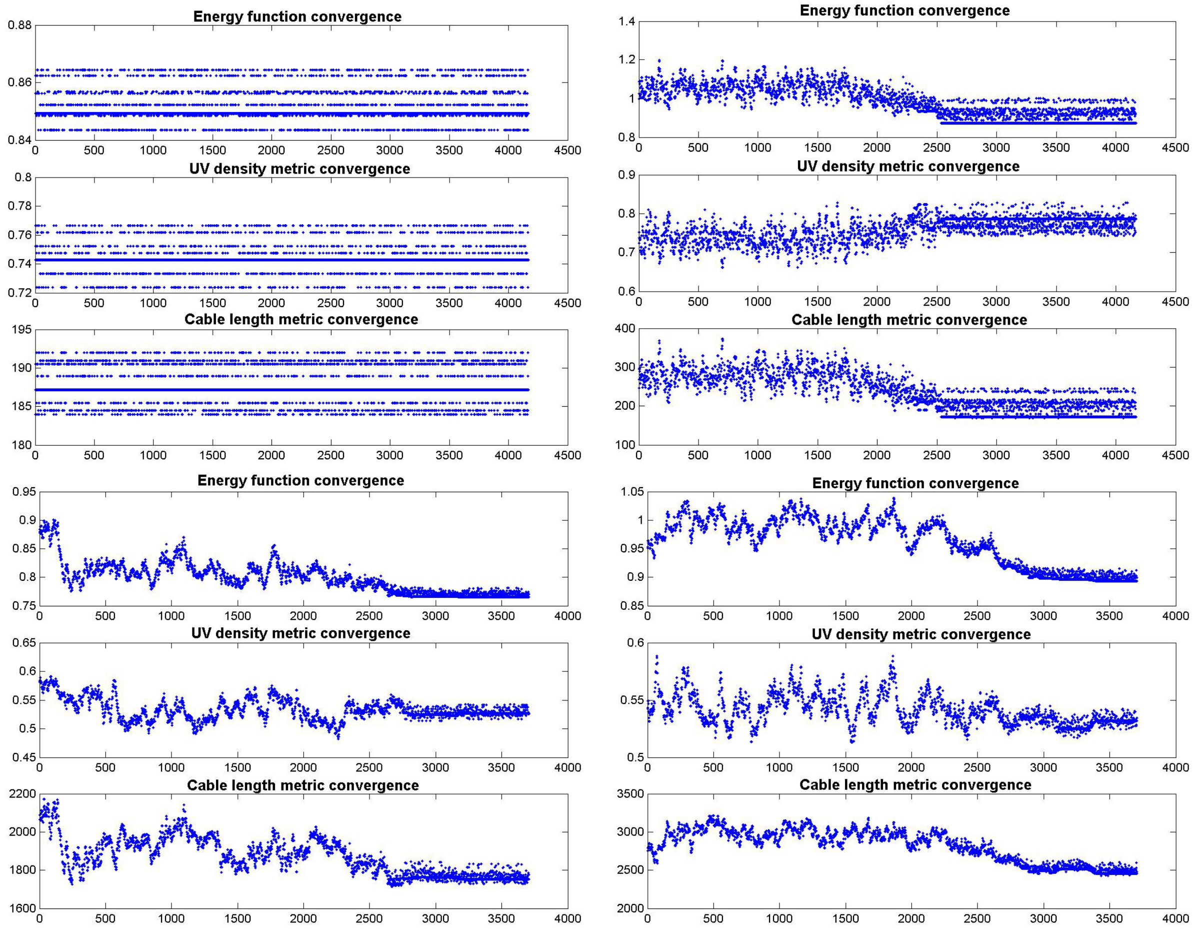


Figure A-25: Energy, uv density and cable length convergence histories; top: $(6, 15, 24)$ stations, bottom $(27, 60, 99)$ stations; left: backward path, right: forward path; random seeds

A.3 Geometries

Non-dominated solutions is another name for Pareto solutions. The geometries for different initially seeded geometries can give insights into the relation between converged Pareto geometries and regular geometries. The figures in this section present the non-dominated shapes (geometries) for all seeded geometries and for both small $(6, 15, 24)$ and large $(27, 60, 99)$ number of stations. Each plot gives the shapes generated with backward (left) and forward (right) approaches. In blue are the first-stage stations, in red the (ones *added*) second-stage stations. Notice that in the backward staging case, the red are the initially optimized, the blue are the *optimally knocked-out* stations.

A.3.1 Non-dominated geometries, $(6, 15, 24)$ stations

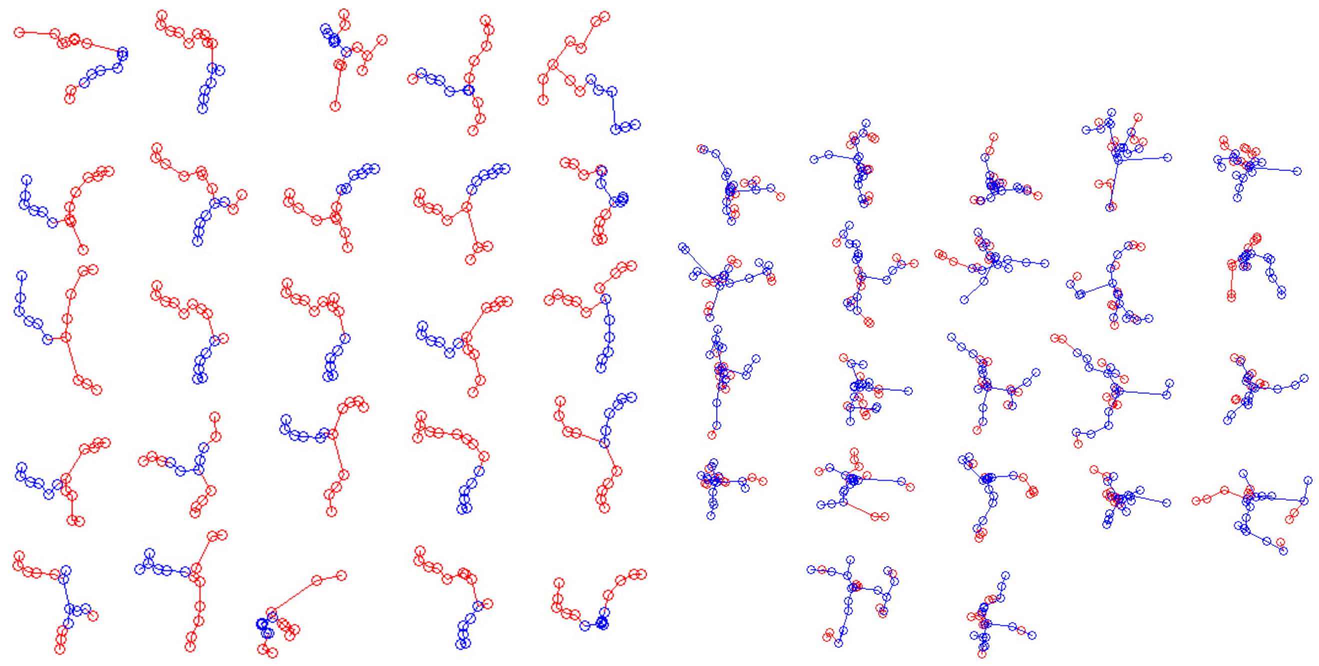


Figure A-26: Non-dominated solutions for backward (left) and forward (right) path, 6 to 15 stations with **random** initial seeds

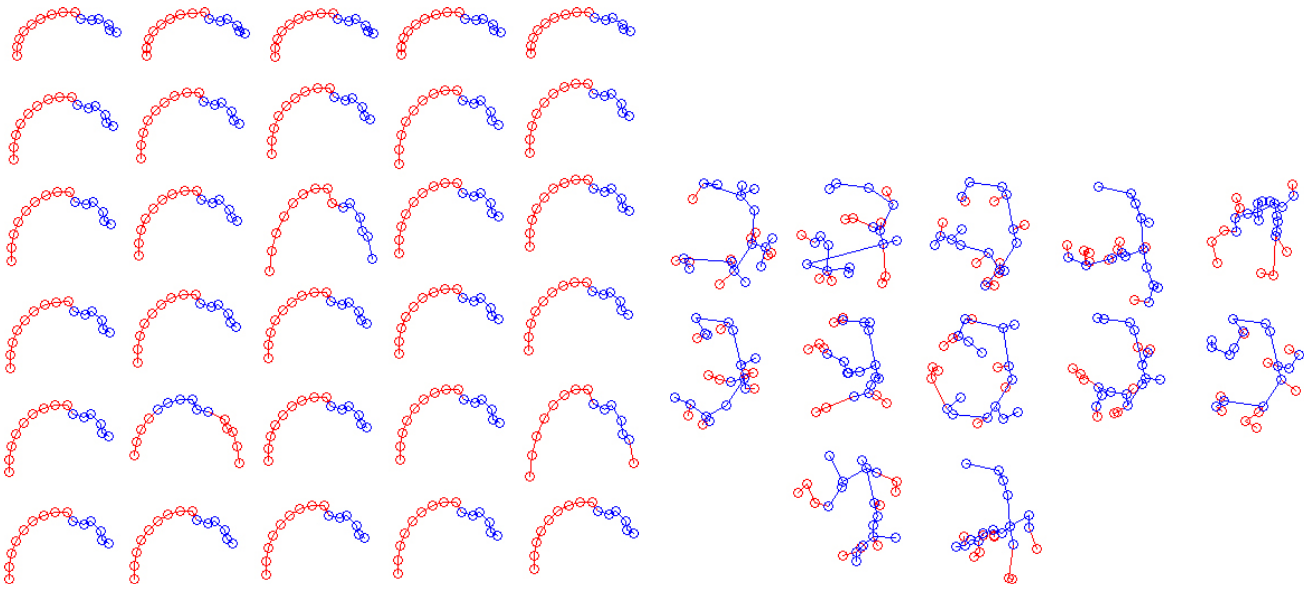


Figure A-27: Non-dominated solutions for backward (left) and forward (right) path, 6 to 15 stations with **circular** initial seeds

A.3.2 Non-dominated geometries, $(27, 60, 99)$ stations

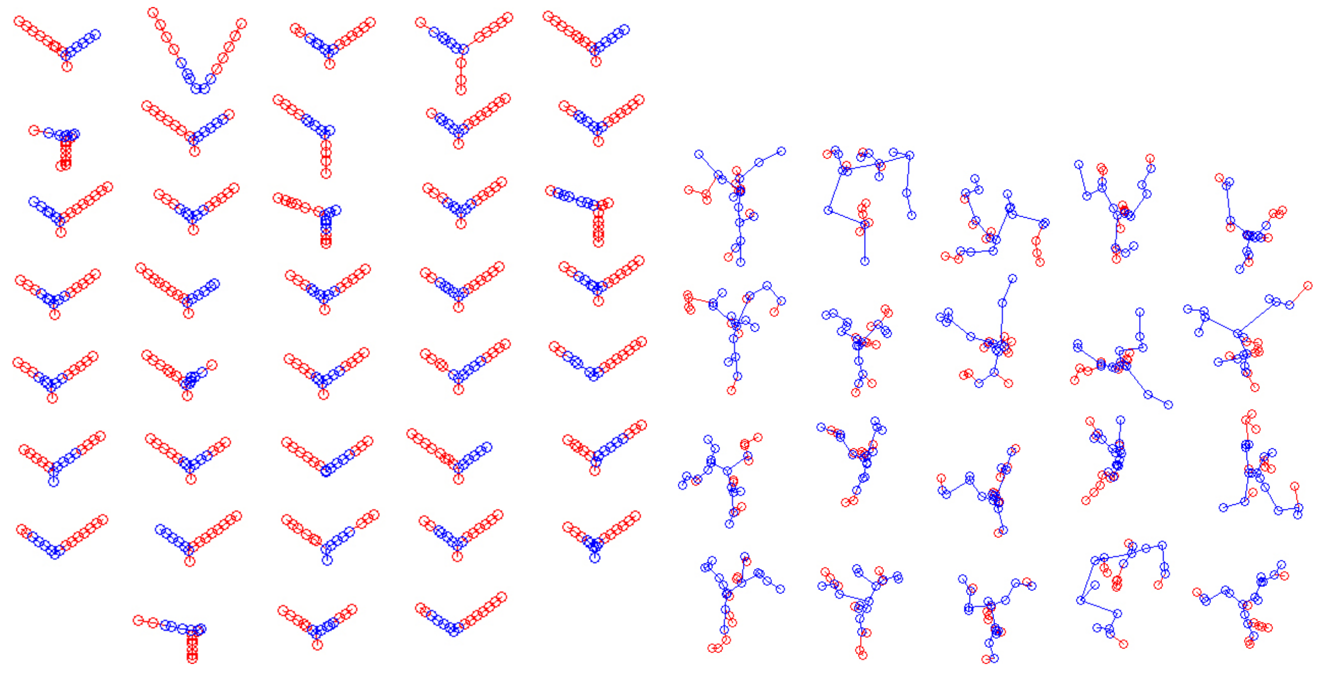


Figure A-28: Non-dominated solutions for backward (left) and forward (right) path, 6 to 15 stations with **VLA-like** initial seeds

A.3.3 Nadir-point solutions

The nadir point is the point closest to the utopia point on the Pareto front. Often this point is considered the *best* design. In this section, we show the nadir-point geometries for all initial seeds and their evolution across stages, for both small and large number of stations.

$(6, 15, 24)$ ***stations***

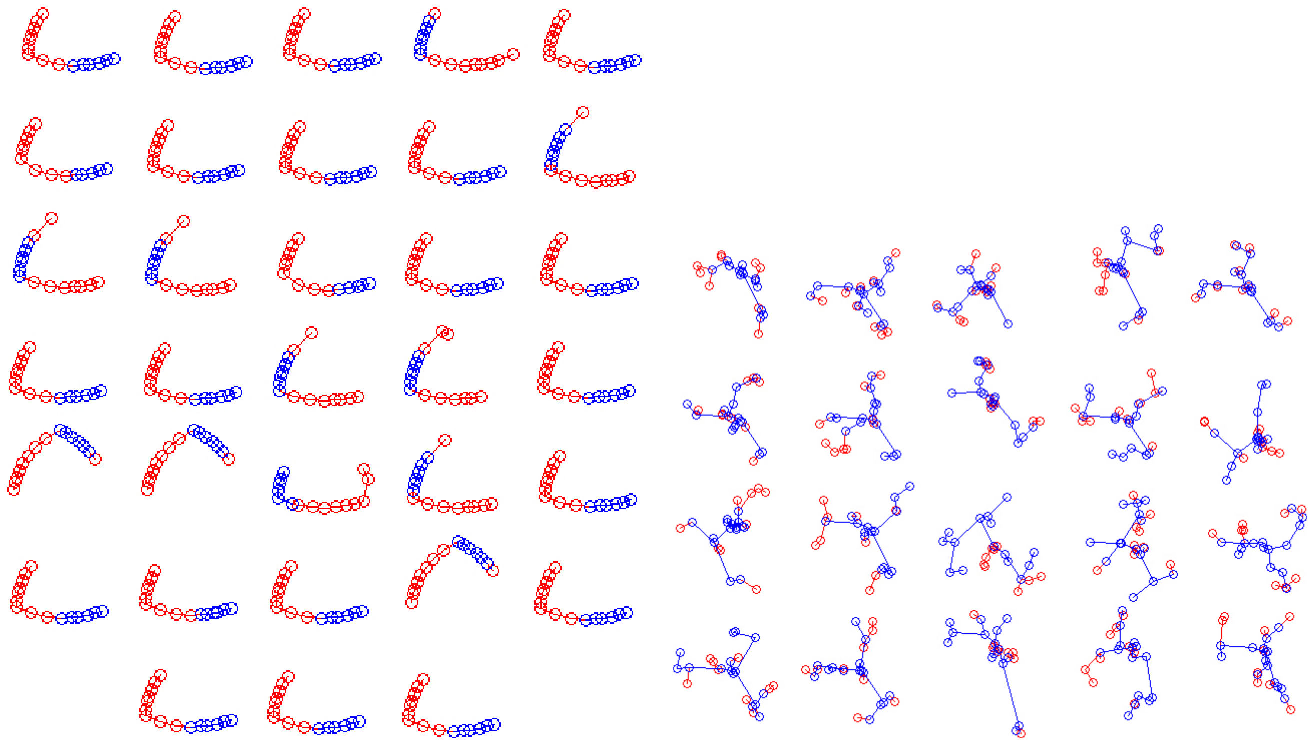


Figure A-29: Non-dominated solutions for backward (left) and forward (right) path, 6 to 15 stations with **Realeaux triangular** initial seeds

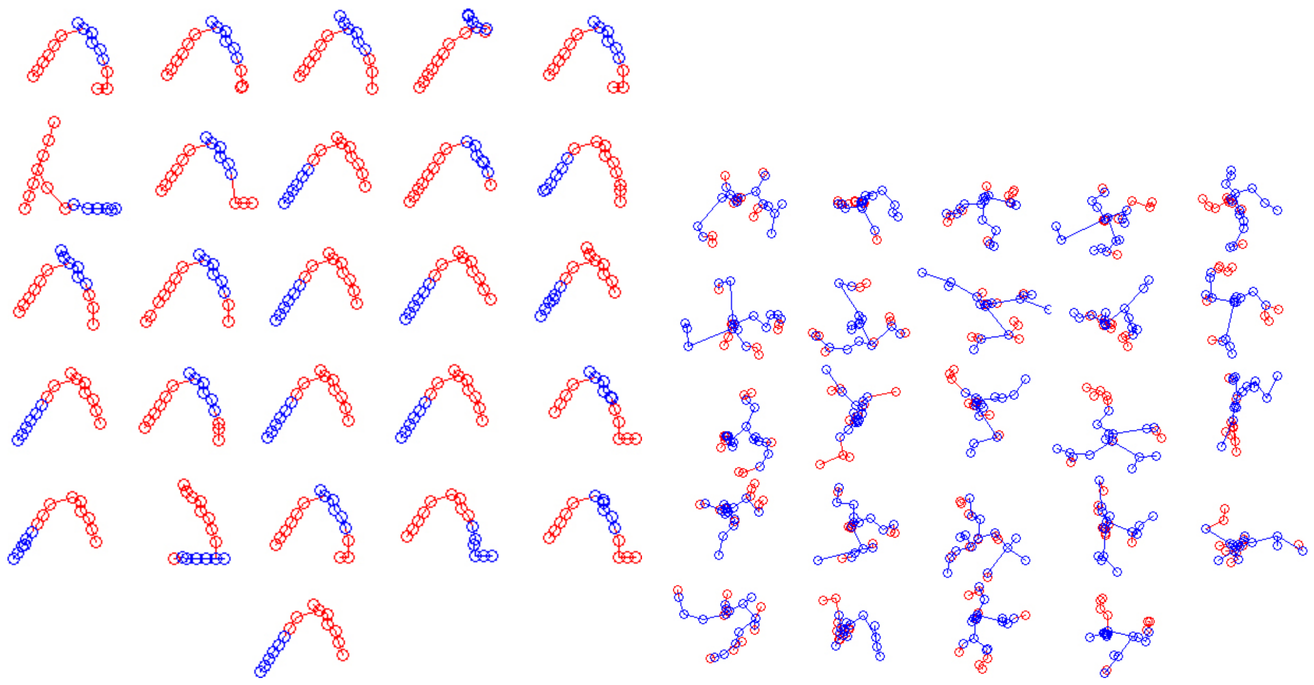


Figure A-30: Non-dominated solutions for backward (left) and forward (right) path, 6 to 15 stations with **triangular** initial seeds

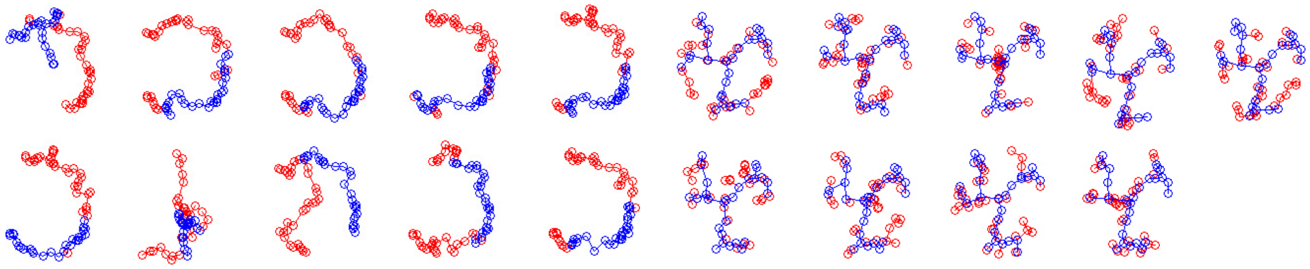


Figure A-31: Non-dominated solutions for backward (left) and forward (right) path, 27 to 60 stations with **random** initial seeds

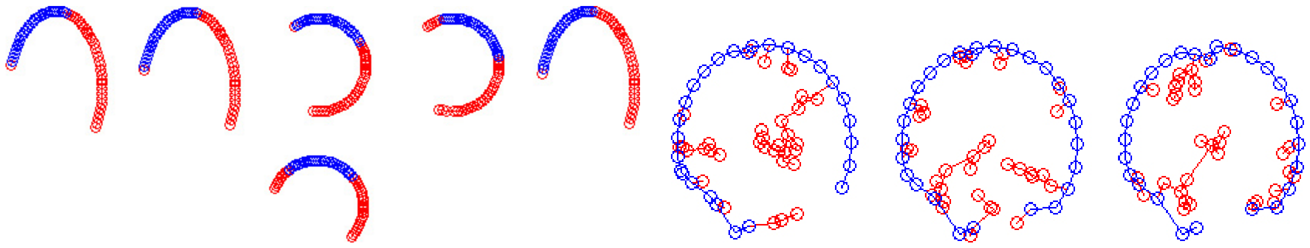


Figure A-32: Non-dominated solutions for backward (left) and forward (right) path, 27 to 60 stations with **circular** initial seeds

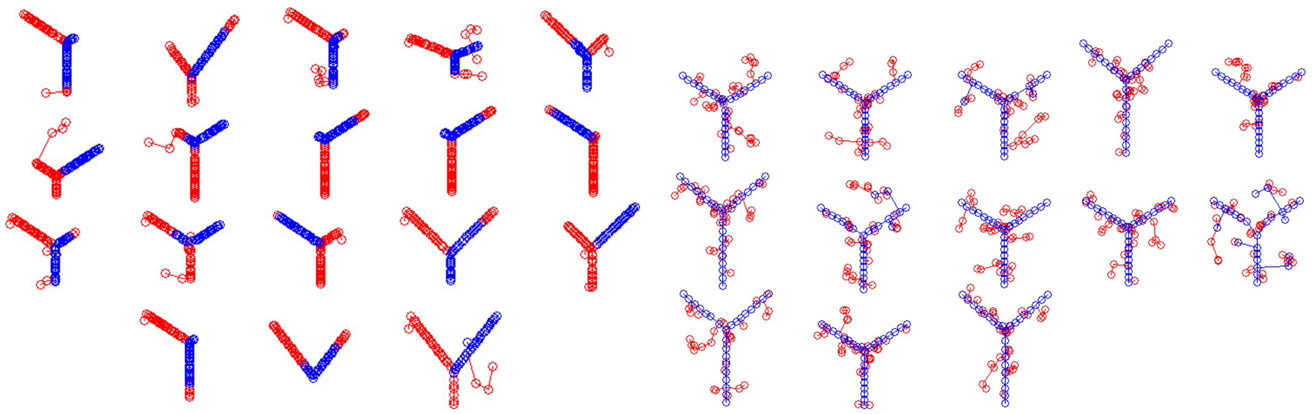


Figure A-33: Non-dominated solutions for backward (left) and forward (right) path, 27 to 60 stations with **VLA-like** initial seeds

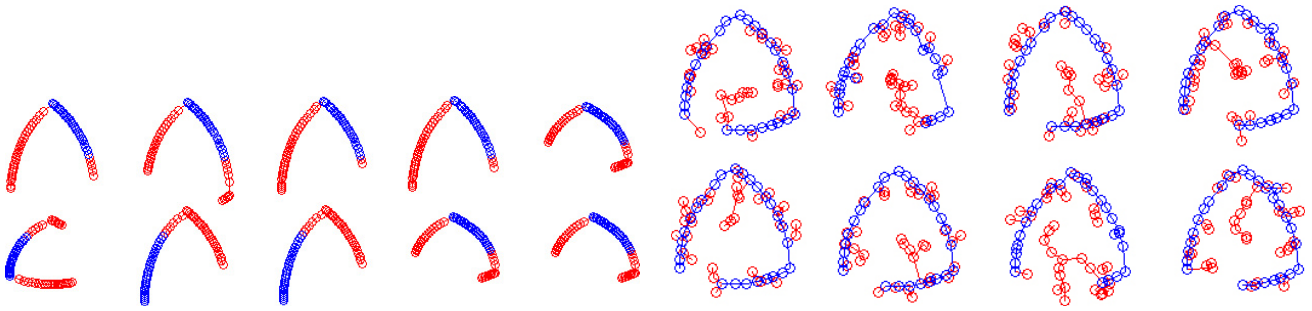


Figure A-34: Non-dominated solutions for backward (left) and forward (right) path, 27 to 60 stations with **Realeaux triangular** initial seeds

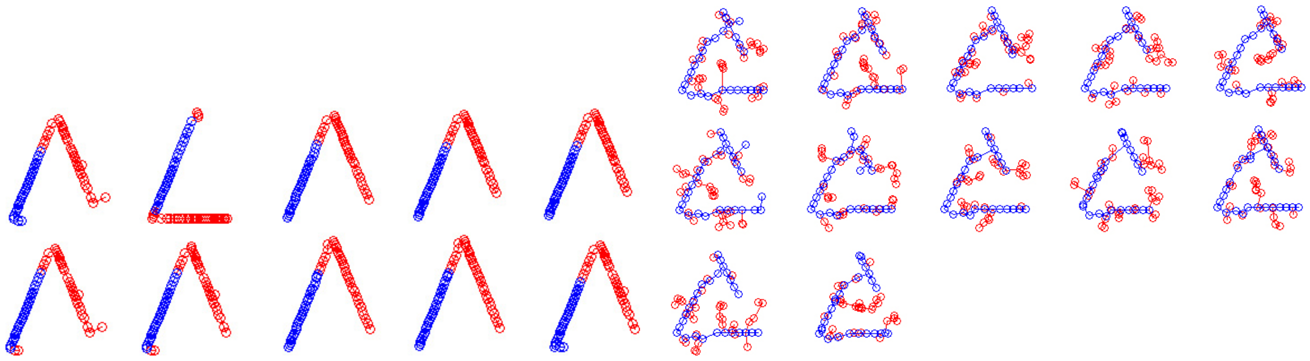


Figure A-35: Non-dominated solutions for backward (left) and forward (right) path, 27 to 60 stations with **triangular** initial seeds

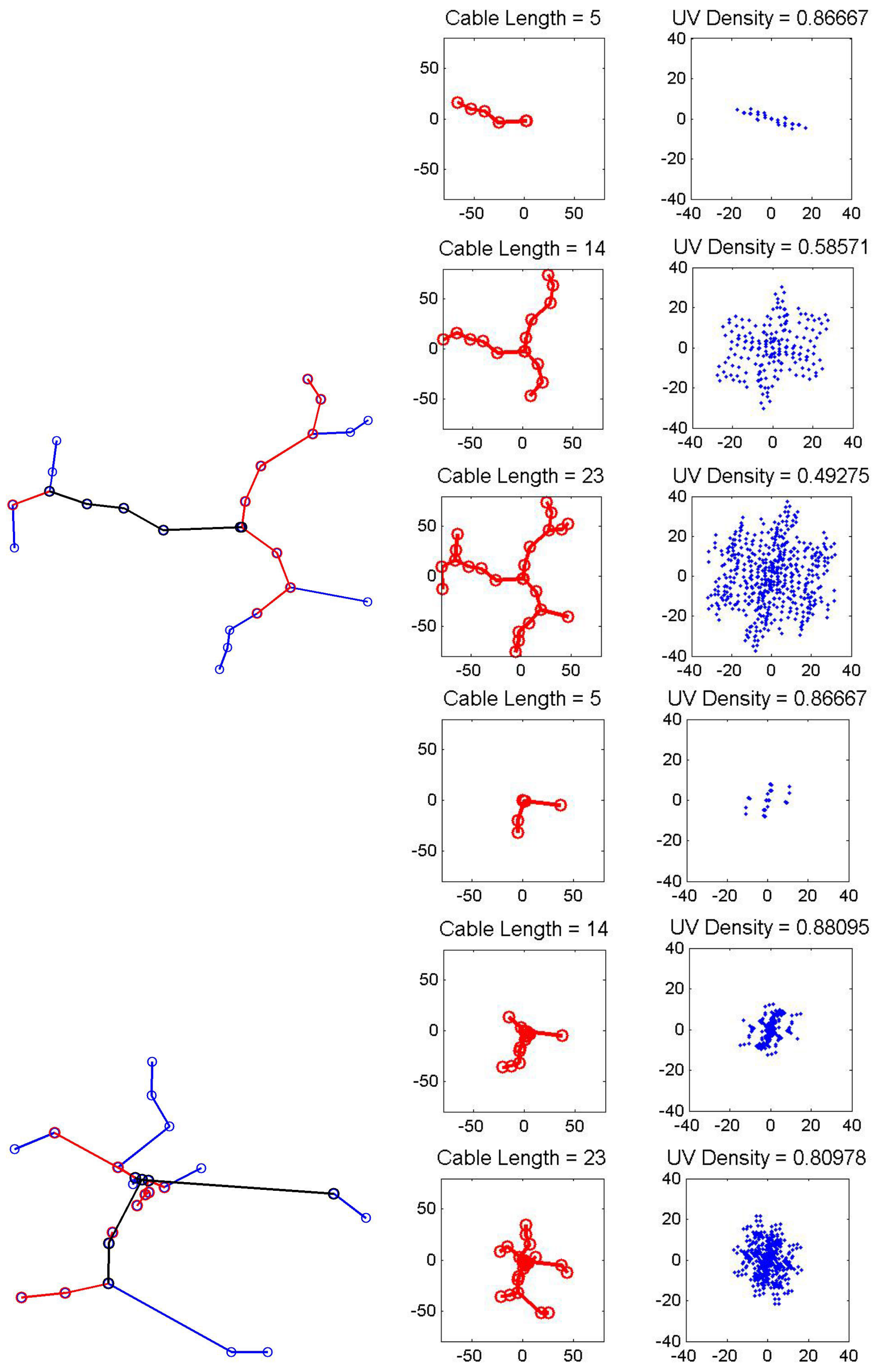


Figure A-36: Nadir point geometries, random seeds; top: backward, bottom: forward

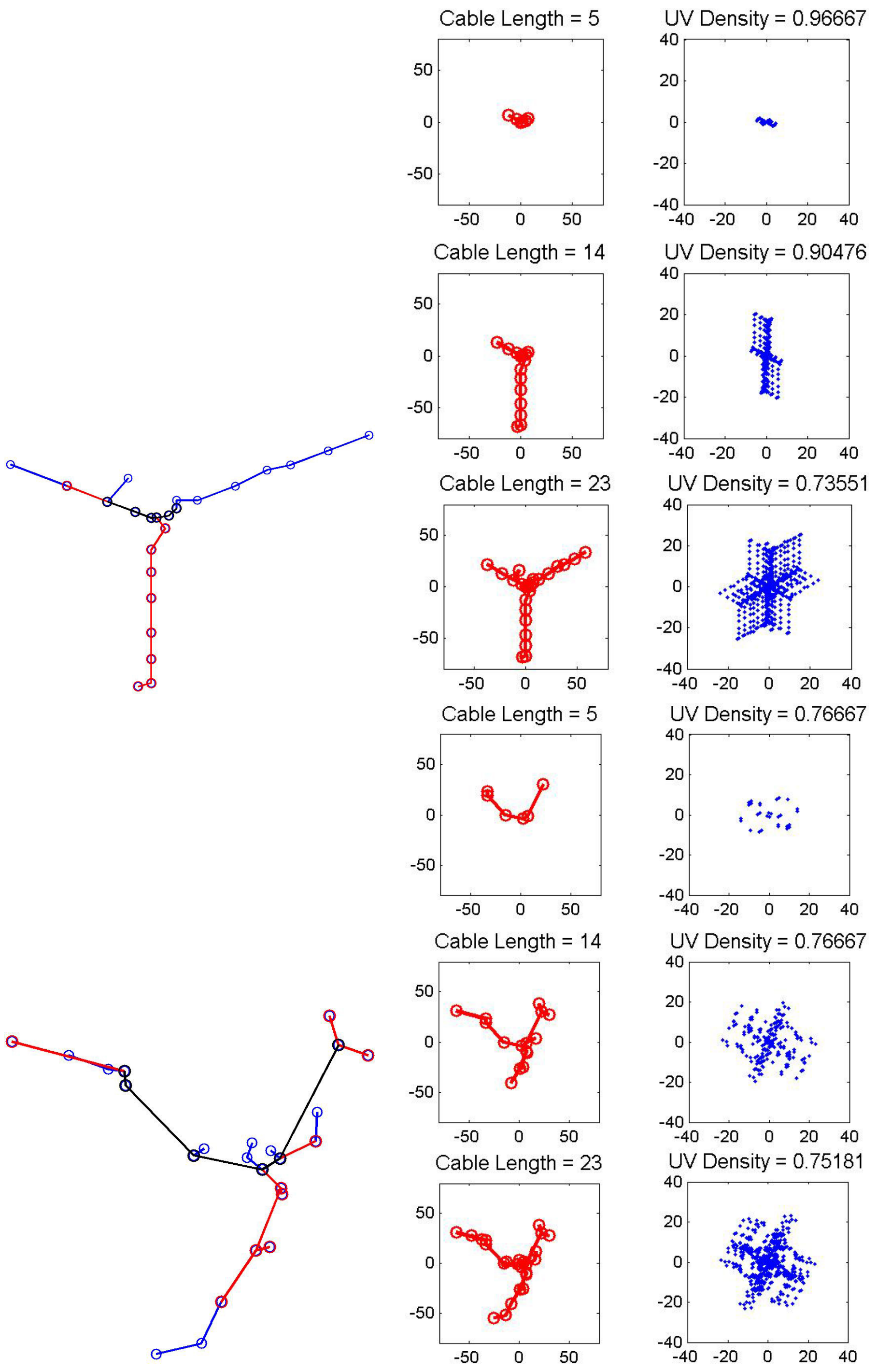


Figure A-37: Nadir point geometries, VLA-like seeds; top: backward, bottom: forward

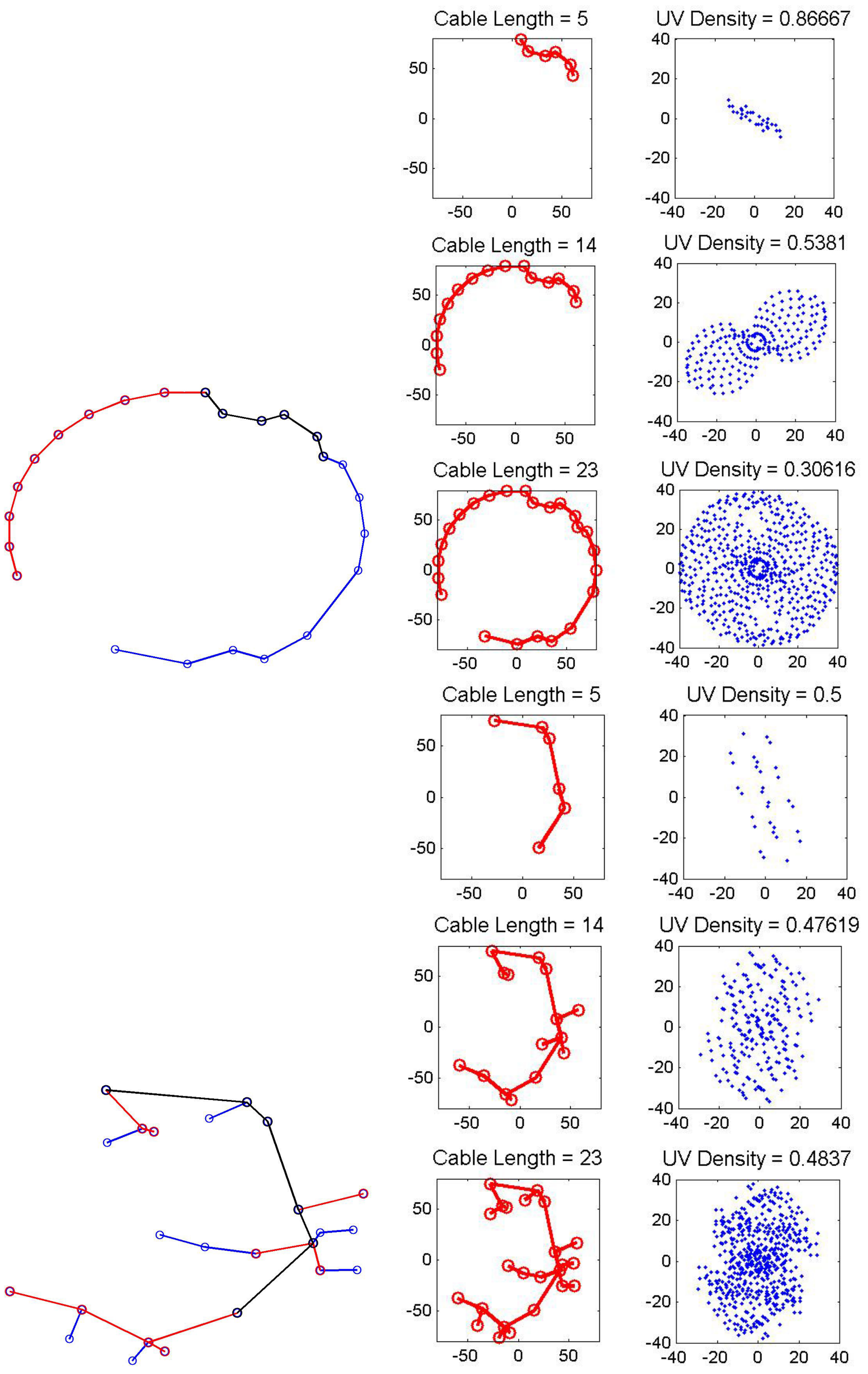


Figure A-38: Nadir point geometries, circular seeds; top: backward, bottom: forward

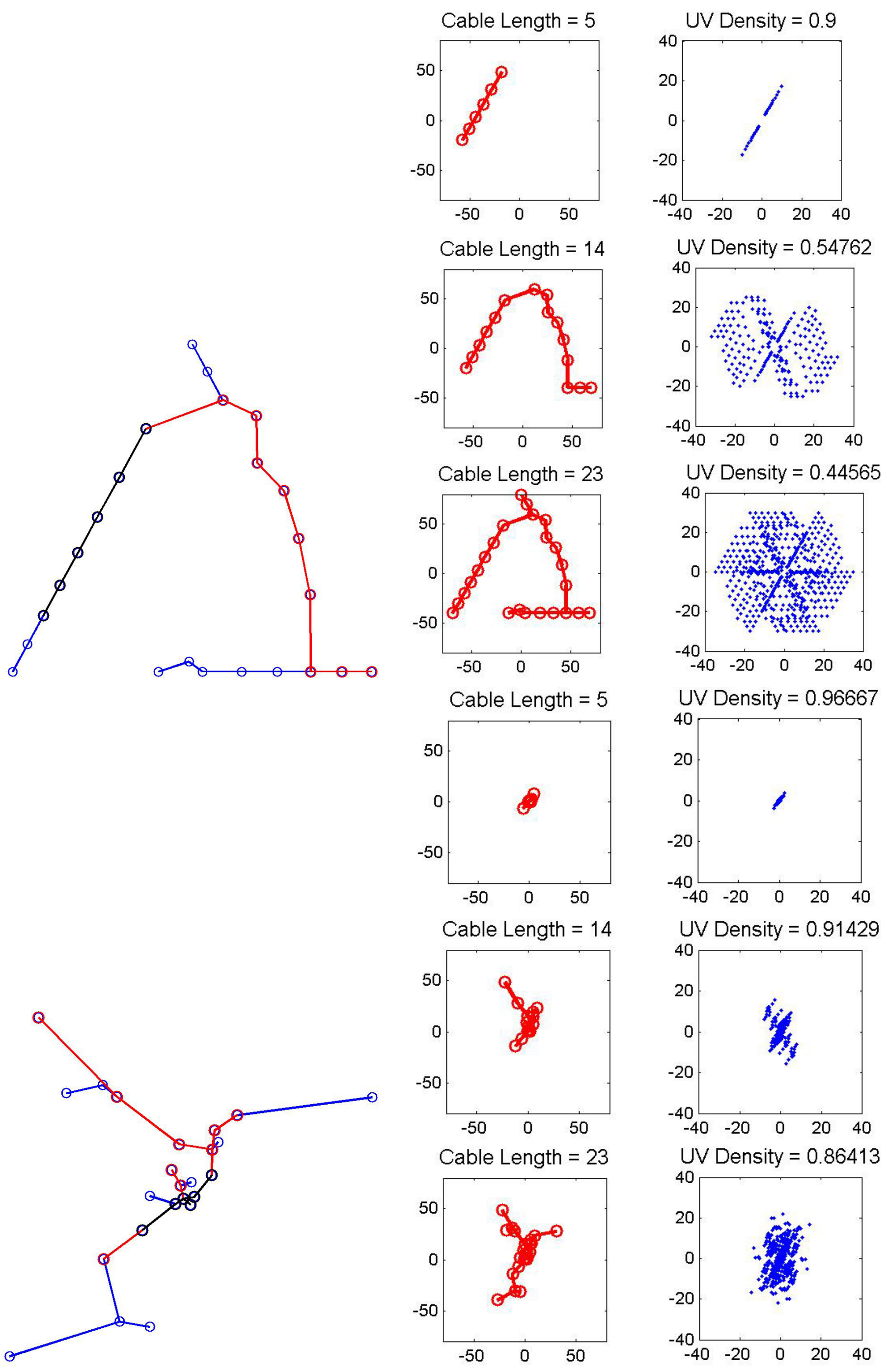


Figure A-39: Nadir point geometries, triangular seeds; top: backward, bottom: forward

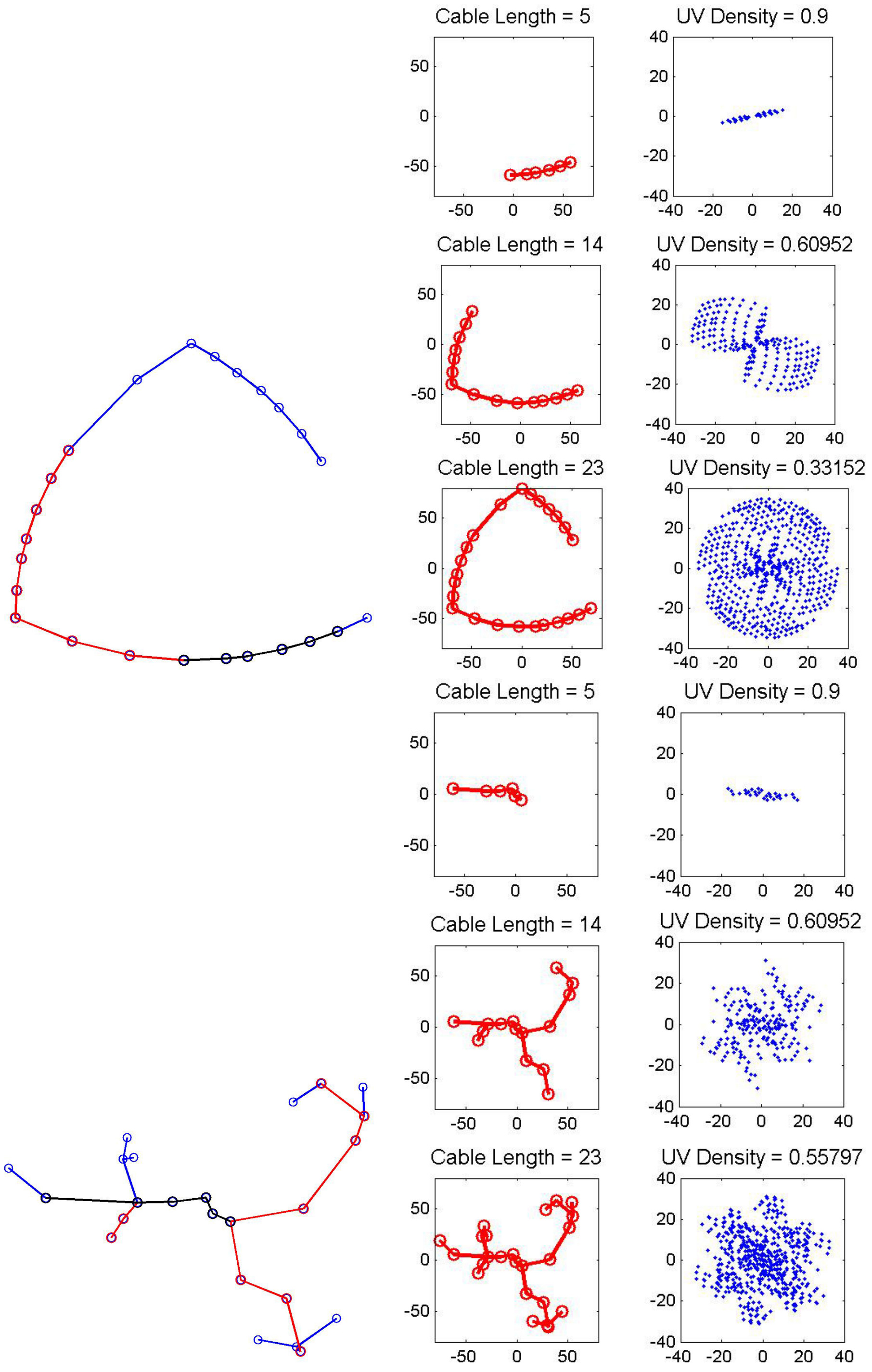


Figure A-40: Nadir point geometries, Reuleaux triangular seeds; top: backward, bottom: forward

$(27, 60, 99)$ ***stations***

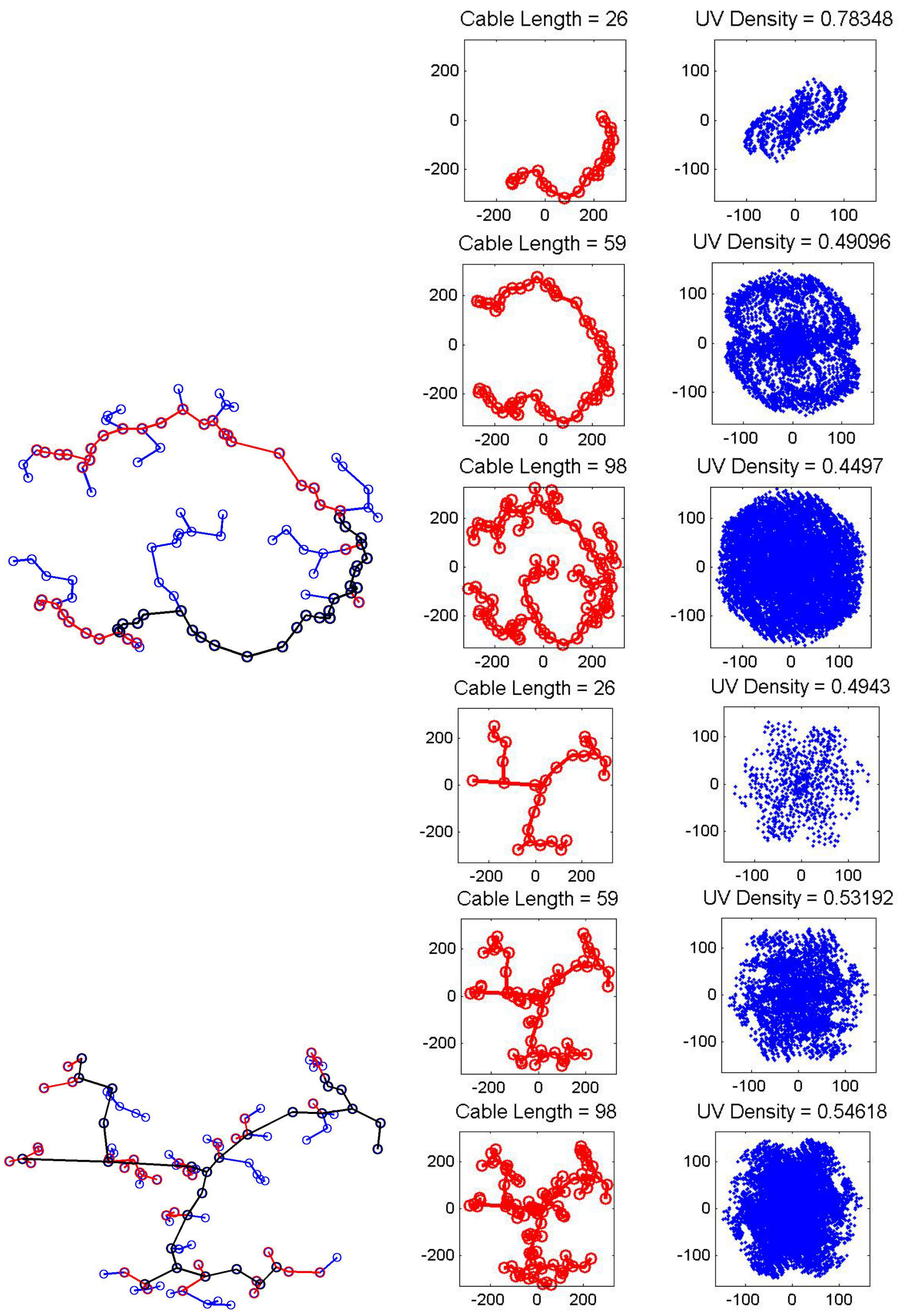


Figure A-41: Nadir point geometries, random seeds; top: backward, bottom: forward

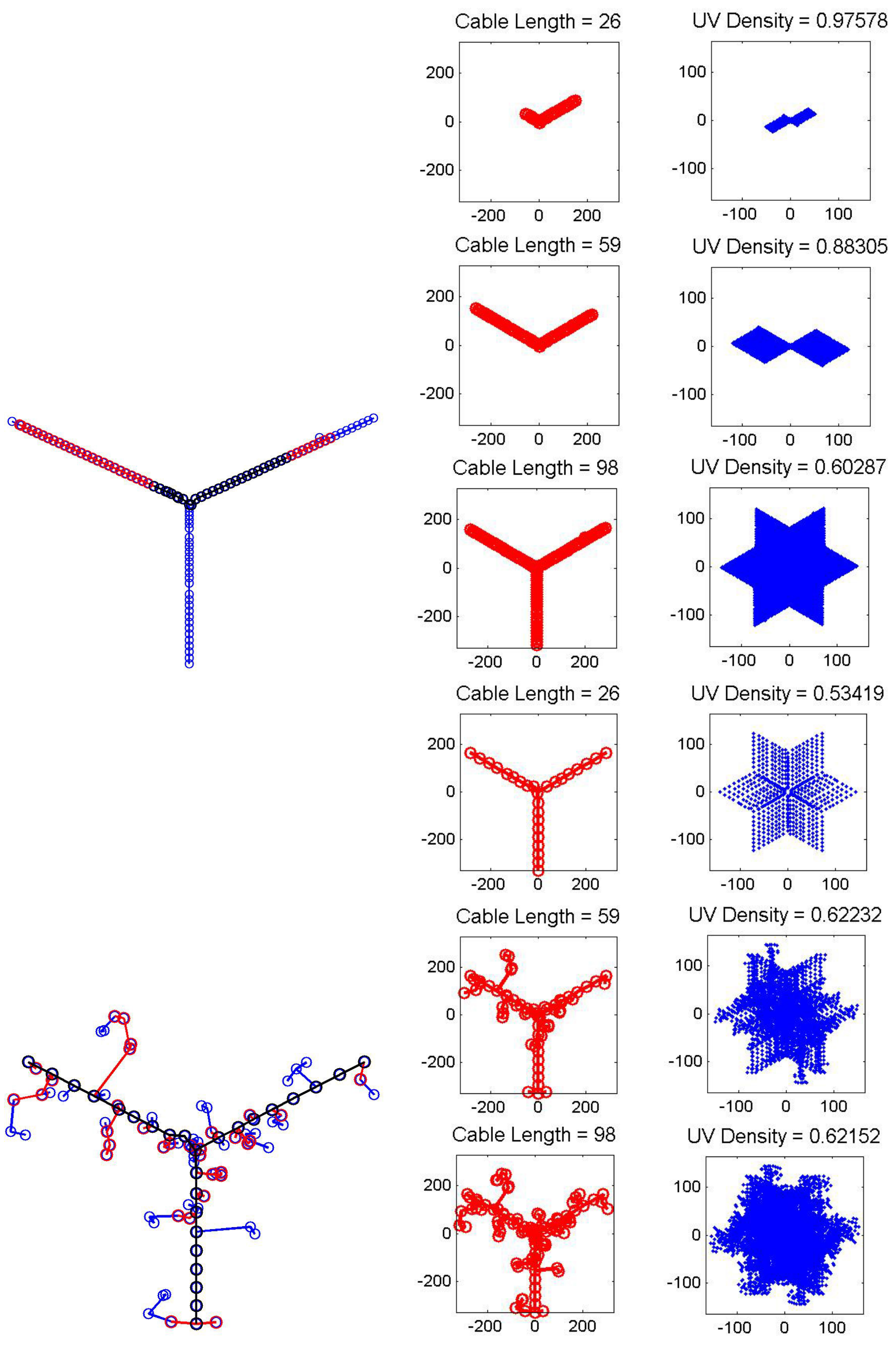


Figure A-42: Nadir point geometries, VLA-like seeds; top: backward, bottom: forward

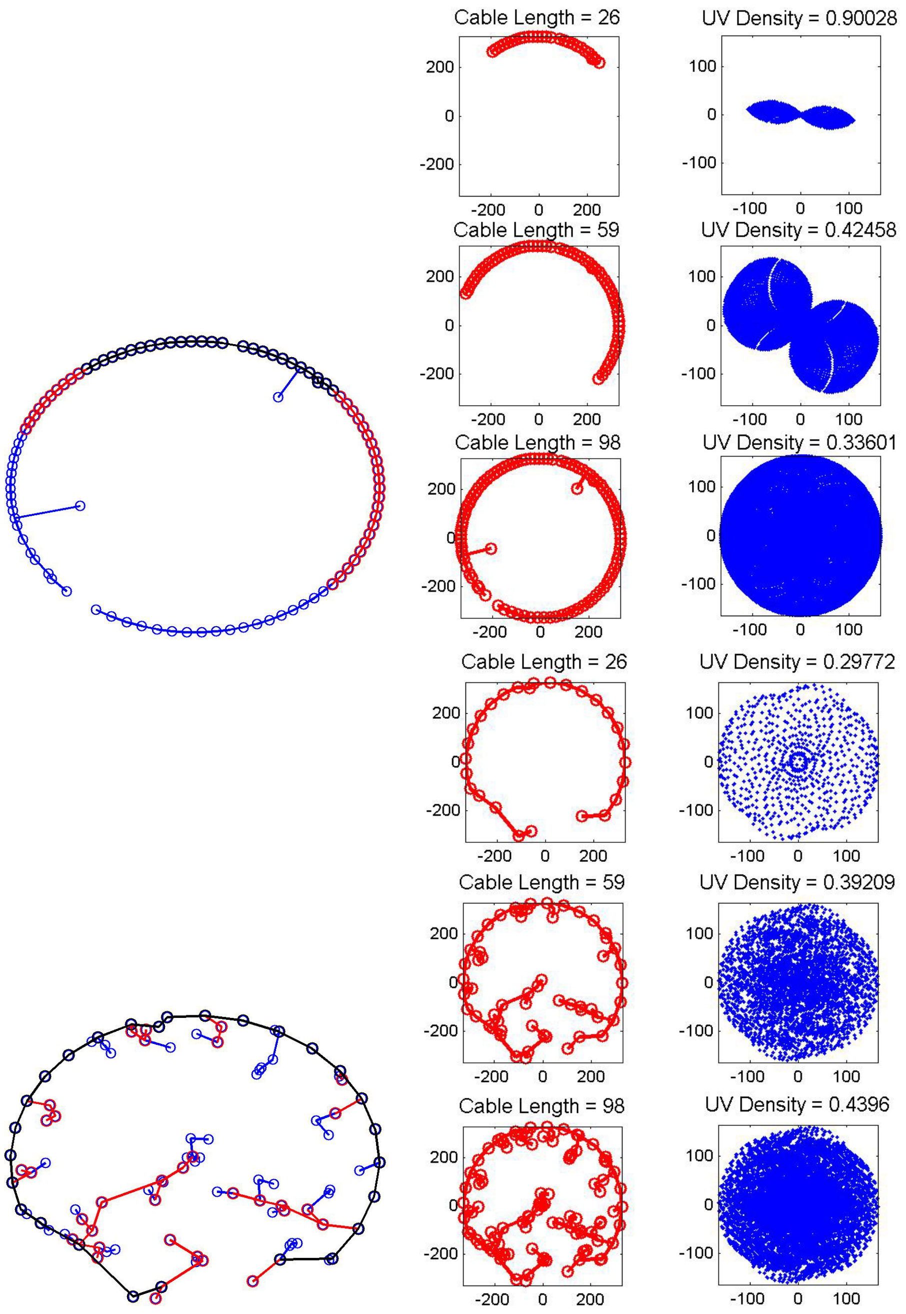


Figure A-43: Nadir point geometries, circular seeds; top: backward, bottom: forward

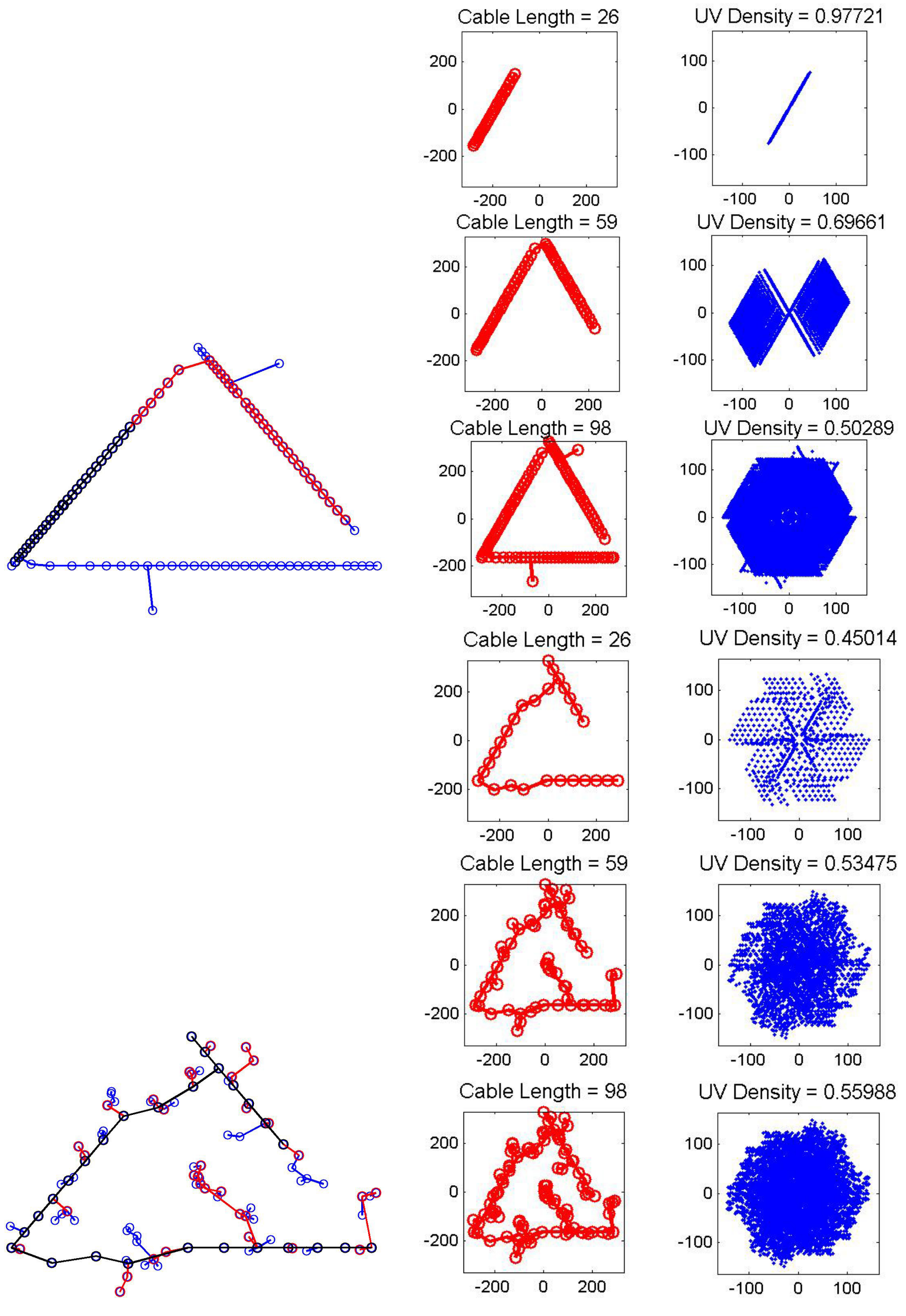


Figure A-44: Nadir point geometries, triangular seeds; top: backward, bottom: forward

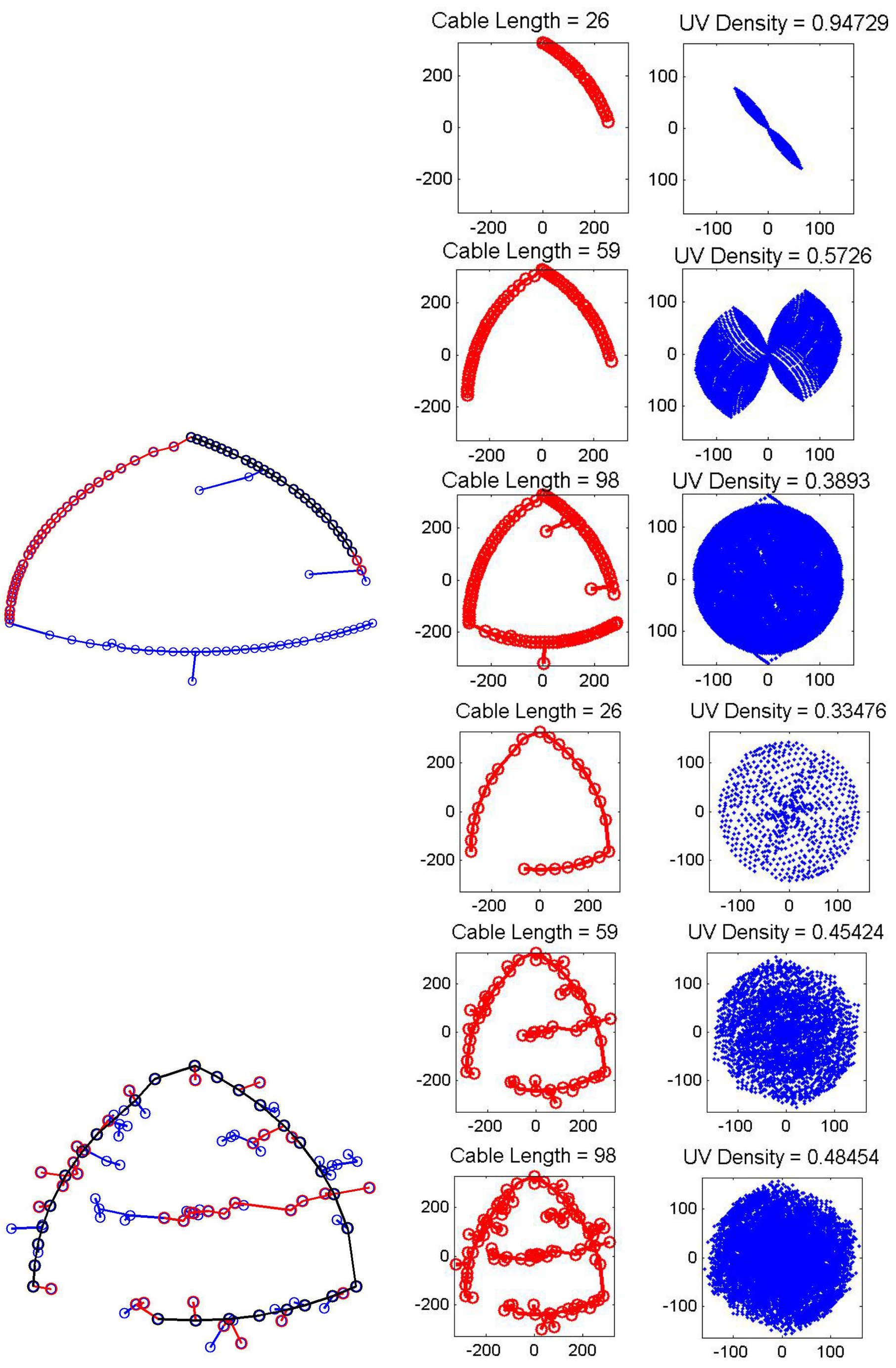


Figure A-45: Nadir point geometries, Reuleaux triangular seeds; top: backward, bottom: forward

A.4 Pareto Front Geometries

Combined Pareto fronts can demonstrate the fitness of certain geometries versus others. Figure A-46 shows results with backward staging for 1st (left) and 2nd stage and for $(6, 15, 24)$ (top) and $(27, 60, 99)$ stations. Figure A-47 shows the same for forward path. Random seeded geometries are plotted in black, circular in blue, VLA-like in red, triangular in green and Reuleaux triangular in magenta.

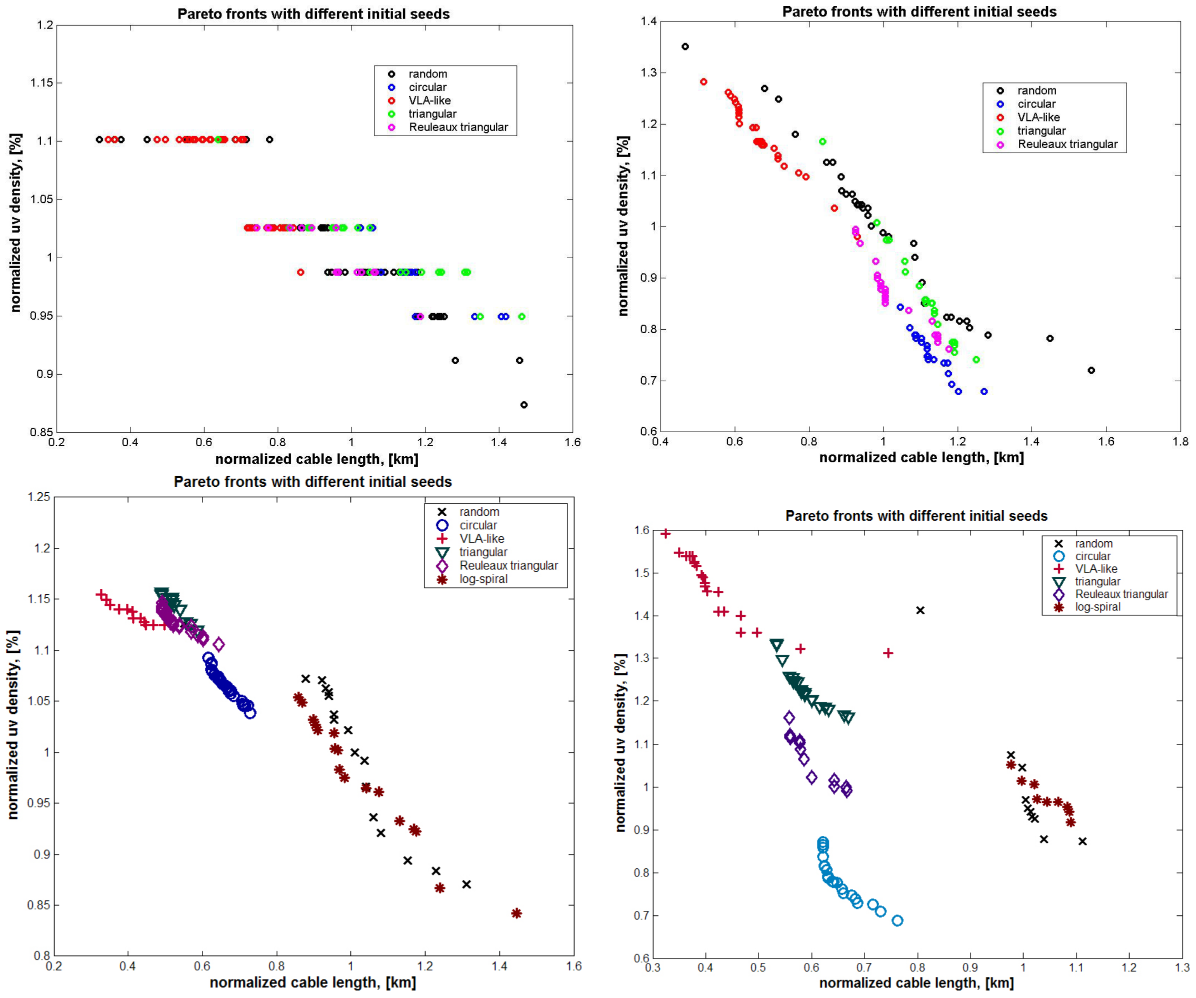


Figure A-46: Backward path, combined Pareto fronts; top: $(6, 15, 24)$ stations, bottom: $(27, 60, 99)$ stations; left: first stage $(15 \rightarrow 6)$ and $(60 \rightarrow 27)$, right: second stage $(24 \rightarrow 15)$ and $(99 \rightarrow 60)$

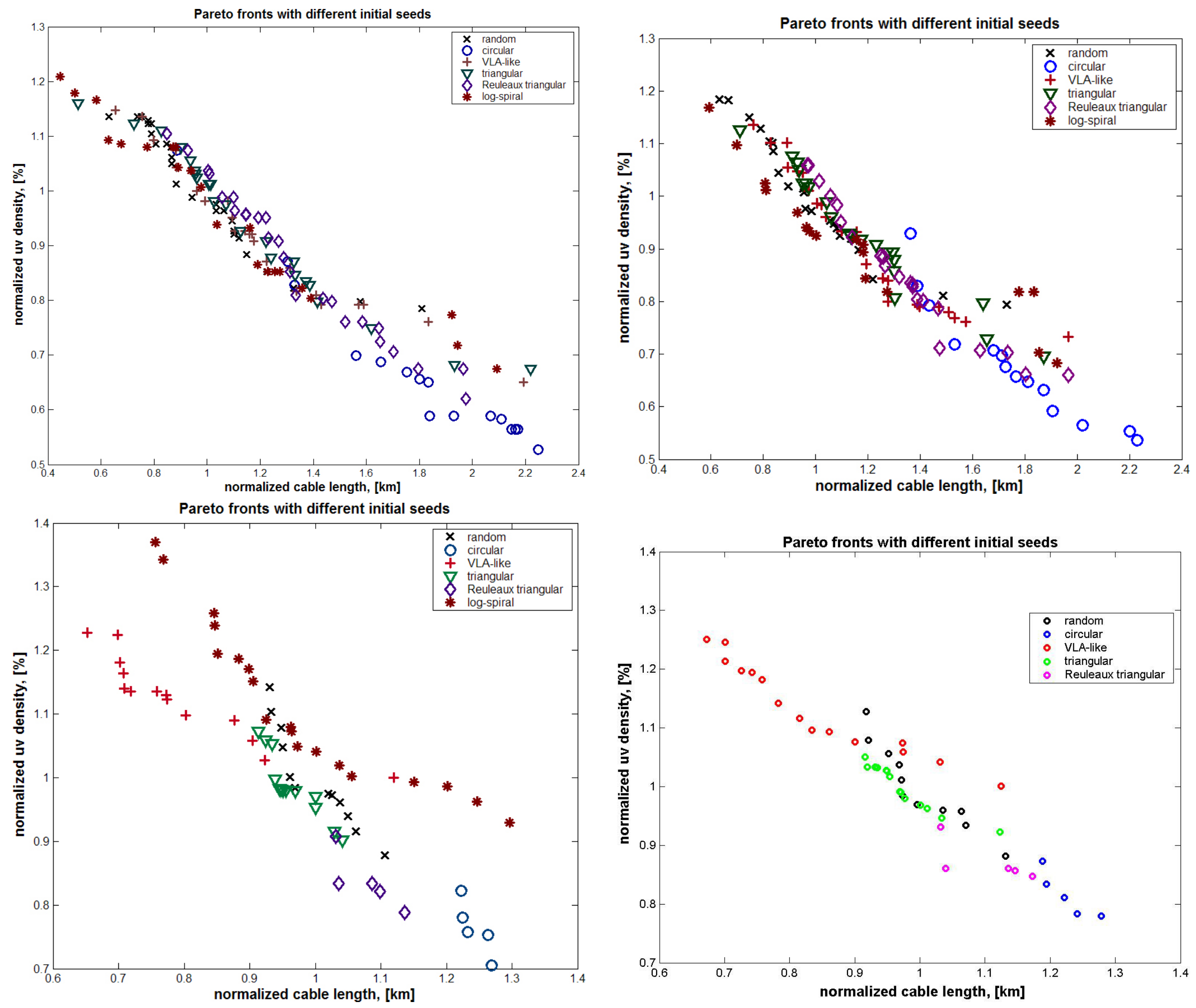


Figure A-47: Forward path, combined Pareto fronts; top: $(6, 15, 24)$ stations, bottom: $(27, 60, 99)$ stations; left: first stage $(6 \rightarrow 15)$ and $(27 \rightarrow 60)$, right: second stage $(15 \rightarrow 24)$ and $(60 \rightarrow 99)$

Appendix B

Selected Algorithms

B.1 Shortest Path Algorithm

```
% Mean average path algorithm:
% Implements shortest path algorithm, from a single vertex to all others
% Gergana Bounova, Last Updated: December 13, 2004

function [dd,d] = shortest_path(adj,adj_length,ind1,ind2)

% need set of verteces and connecting edges
stations = size(adj,2);

d = inf*ones(1,stations); d(ind1) = 0;

% form V set
V = ind1; index = ind1;

while not(isempty(V))
% ------------------------------
for i=1:size(V)
    % identify V member
    ind = V(i);
    for j=1:stations
        if adj(ind,j)==1 & d(j)>d(ind)+adj_length(ind,j)
            d(j)=d(ind)+adj_length(ind,j);
            % remove node ind and add node j
            exists = 0; % ------------------
            for c=1:size(index)
                if index(c)==j
                    'node already exists';
                    exists = 1;
                end
            end
            if not(exists)
                V = [V; j];
                index = [index; j];
            end % --------------------------
        end
    end
    if length(index)>=stations
```

```
        break;
    end
end V = V(2:size(V));
% -------------------------------
end

dd = d(ind2);
```

B.2 Minimum linkage algorithm

```
% new cabling routine for forward staging
% Gergana Bounova, Last Updated: May 16, 2004

function [cable,adj2wb,adj_lengths2wb] =
newmincable(st1,st2,adj1,adj_lengths1)

num_stations(1) = length(st1); num_stations(2) = length(st2);
adj2wb = zeros(length(st2)); adj_lengths2wb = adj2wb; indeces1 =
zeros(num_stations(1),1); for i=1:num_stations(1)
    st11 = squeeze(st1(:,i));
    for j=1:num_stations(2)
        st22 = squeeze(st2(:,j));
        if norm(st11-st22)<0.1
            indeces1(i) = j;
        end
    end
end

for i=1:num_stations(1)
    for j=1:num_stations(1)
        adj2wb(indeces1(i),indeces1(j))=adj1(i,j);
        adj_lengths2wb(indeces1(i),indeces1(j))=adj_lengths1(i,j);
    end
end

station_stack = squeeze(st1(:,:)); for i=1:num_stations(2)
    st = [st2(1,i) st2(2,i)];
    d = ones(length(station_stack),1)*st-station_stack';
    dn = zeros(length(station_stack),1);
    for j=1:length(dn)
        dn(j) = norm(d(j,:));
    end
    if min(dn)<0.01
        'do nothing';
    else
        ind = pdesubixe(dn,min(dn));
        adj2wb(i,ind) = 1;
        adj2wb(ind,i) = 1;
        adj_lengths2wb(i,ind) = norm(st'-station_stack(:,ind));
        adj_lengths2wb(ind,i) = adj_lengths2wb(i,ind);
        station_stack = [station_stack st'];
    end
end
```

```
cable = sum(sum(adj_lengths2wb))/2;
```

B.3 Backward DOE Algorithm

```
% DOE for backward staging path
% Gergana Bounova, Last Updated: June 18, 2004

% input: stations2 = second-stage array + adjacency and adj-lengths
%                    matrices
% output: stations1 = first-stage array + adjacency and adj_lengths
%                    matrices
% s - stage number

for i=1:num_stations(s+1)
  node(i).child = [];
  for j=1:num_stations(s+1)
    if adj2d(i,j)==1
      node(i).child = [node(i).child; j];
    end
  end
end

% ==============================================================
stations1_sample = zeros(num_stations(s+1),2,num_stations(s));
adj1_sample =
zeros(num_stations(s+1),num_stations(s),num_stations(s));
adj_lengths1_sample = adj1_sample; indeces1_sample =
zeros(num_stations(s+1),num_stations(s));

for ss=1:num_stations(s+1)
  inds = 1;
  stations1(:,inds)=stations2(:,ss);
  indeces1(inds) = ss;

  while inds < num_stations(s)
    [y,ind]=sort(node(indeces1(inds)).child); % sort children of
                                              % first node
    curr = inds;
    for x=1:length(y)
      new = node(indeces1(curr)).child(ind(length(y)-x+1));
      if belongsto(new,indeces1(1:inds))
        'do nothing';
        allvisited = 1;
      else
        allvisited = 0;
        inds = inds + 1;
        indeces1(inds) = new;
        break;
      end
    end

    if allvisited
```

```
      % jump somewhere else?
      found = 0;
      for xx=1:inds
        for l=1:length(node(indeces1(xx)).child)
          new = node(indeces1(xx)).child(l);
          if not(belongsto(new,indeces1(1:inds)))
            inds = inds + 1;
            indeces1(inds) = new;
            found = 1;
            break;
          end
        end
        if found
          break;
        end
      end
    end

  end
  indeces1_sample(ss,:) = indeces1;
  stations1 = stations2(:,indeces1);
  stations1_sample(ss,:,:) = stations1;

end

for ss=1:num_stations(s+1)
  adj1_sample(ss,:,:)=adj2d(indeces1_sample(ss,:),indeces1_sample(ss,:));
  adj_lengths1_sample(ss,:,:)=adj_lengths2d(indeces1_sample(ss,:),indeces1_sample(ss,:));
end

% pick the best configuration
[xgrid,ygrid,TRI]=get_grid(num_stations(s),xygrid,wavelength,uvscale);
for ss=1:length(adj_lengths1_sample)
    x = squeeze(stations1_sample(ss,1,:)); y = squeeze(stations1_sample(ss,2,:));
    dns(ss) = uvdens_circ_eq_area(x,y,num_stations(s),wavelength,xgrid,ygrid,TRI);
    cbl(ss) = sum(sum(squeeze(adj_lengths1_sample(ss,:,:))))/2;
end distance = (dns/mean(dns)).^2 + (cbl/mean(cbl)).^2;
[mindist,ind] = min(distance); stations1 =
squeeze(stations1_sample(ind,:,:)); indeces1 =
squeeze(indeces1_sample(ind,:)); adj1d =
squeeze(adj1_sample(ind,:,:)); adj_lengths1d =
squeeze(adj_lengths1_sample(ind,:,:));
```

B.4 Forward DOE Algorithm

```
% DOE for forward path algorithm
% Gergana Bounova, Last Updated: October 2, 2004

% input: stations1: original first-stage array
% output: stations3: additional stations to stations1 to complete
%                    the second-stage array
% s - stage number
% num_stations - vector of number of stations for each stage
% xys - sample of randomly-generated stations (design space)


stations3_sample =
zeros(num_stations(s),2,num_stations(s)-num_stations(s-1)); count1
= 0;

for ss=1:length(stations3_sample)
    cnt = 0;
    stations3 = zeros(2,num_stations(s)-num_stations(s-1));
    while cnt < (num_stations(s)-num_stations(s-1))
        ind = ceil(rand*length(xys));
        cnt = cnt + 1;
        stations3(:,cnt) = xys(:,ind);
        if cnt> 1 & belongsto(stations3(:,cnt),stations3(:,1:cnt-1))
            cnt = cnt - 1;
        end
    end
    stations3_sample(ss,:,:) = stations3;
end

%% Pick the *best* point
[xgrid,ygrid,TRI]=get_grid(num_stations(s),xygrid,wavelength,uvscale);
for ss=1:length(stations3_sample)
    stations2 = [stations1 squeeze(stations3_sample(ss,:,:))];
    x = stations2(1,:); y = stations2(2,:);
    [dns(ss)] = uvdens_circ_eq_area(x,y,length(x),wavelength,xgrid,ygrid,TRI);
    [cbl(ss)] = newmincable(stations1,stations2,adj1d,adj_lengths1d);
end

distance = (dns/min(dns)).^2 + (cbl/min(cbl)).^2;

[mindist,ind] = min(distance); stations3 =
squeeze([stations3_sample(ind,1,:); ...
                    stations3_sample(ind,2,:)]);
```

Bibliography

[1] Webster Online, source: http://www.webster-dictionary.org/

[2] *Wikipedia, the free encyclopedia*, source: http://en.wikipedia.org/

[3] Cohanim, B.E., Hewitt, J.N, de Weck, O.L. *The Design of Radio Telescope Array Configurations using Multiobjective Optimization: Imaging Performance versus Cable Length*, The Astrophysical Journal Supplement Series, 154:705-719, 2004 October

[4] Chaize, M. *Enhancing the Economics of Satellite Constellations via Staged Deployment and Orbital Reconfiguration*, Master's Thesis, MIT, Cambridge, MA 02139, May 2003

[5] Yang, L. and R. Kornfeld, *Examination of the Hub-and-Spoke Network: A Case Example Using Overnight Package Delivery*, in Proceedings of the 41st Aerospace Sciences Meeting and Exhibit, January 6-9, 2003, Reno, Nevada, AIAA 2003-1334.

[6] Rosenberger, J. M., E. L. Johnson, and G. L. Nemhauser. *A Robust Assignment Model with Hub Isolation and Short Cycles.* White papers and reports series, the Logistics Institute, Georgia Institute of Technology, 2001.

[7] Mathias, N., Gopal, V., *Small worlds: How and Why*, Physical Review E, Volume 63, 021117, January 26, 2001

[8] Newman, M.E.J, *The Structure and Function of Complex Networks*, SIAM Review, Volume 45, Number 2, pp.167-256

[9] Bertsekas, D., *Network Optimization Continuous and Discrete Models*, Athena Scientific, Belmont, MA, 1998.

[10] Steiner tree from *MathWorld*, source: http://mathworld.wolfram.com/SteinerTree.html

[11] Barabasi, A-L., Albert,R., *Emergence of scaling in random networks.* Science(286)509-512, 1999

[12] Cohanim, B., *Multiobjective Optimization of a Radio Telescope Array with Site Constraints*, Master's thesis, MIT, Cambridge, MA 02139, Feb 2004.

[13] *Multidisciplinary System Design Optimization*, MIT course 16.888/ESD.77, Spring 2004, Lecture notes, source: http://stellar.mit.edu/S/course/16/sp04/16.888/index.html, Date accessed: 12/20/2004.

[14] LOFAR: http://www.lofar.org

[15] Kirkpatrick, S., Gelatt, C.D.,Vecchi, M.P.*Optimization by Simulated Annealing*, Science, New Series, Vol. 220, No. 4598 (May 13, 1983), 617-680

[16] de Weck, O., *System Optimization with Simulated Annealing (SA),* Memorandum.

[17] Audiovisual Online, source:

`http://www.audiovisualonline.co.uk/dynamic/eshop_products.set/ref/15/display.html`

, Date accessed: January 16, 2005

[18] Doyle, J., Willinger, W., *Robustness and the Internet: Design and Evolution*, March 1, 2002

[19] Barabsi, A-L., Oltvai, Z. Network Biology: Understanding the Cell's Functional Organization, Nature Review: Genetics, Volume 5, February 2004

[20] Watts D., Strogatz S. *Collective dynamics of 'small-world' networks* Nature. 1998 Jun 4;393(6684):440-2

[21] Barabasi, A-L., Albert, R. *Emergence of Scaling in Random Networks*, Science, 286:509-512, October 15, 1999

[22] Callaway D.S, Newman M.E, Strogatz S.H, Watts D.J., *Network robustness and fragility: percolation on random graphs*, Phys Rev Lett. 2000 Dec 18;85(25):5468-71

[23] http://www.vla.nrao.edu/

[24] *Networks / Pajek Program for Large Network Analysis*, source:

`http://vlado.fmf.uni-lj.si/pub/networks/pajek`

[25] Setubal, Meidanis, "Introduction to Computational Molecular Biology", PWS Publishing Company, 1997