NTNU – Trondheim
Norwegian University of
Science and Technology

Optimization of Combined Cycles for Offshore Oil and Gas Installations

Alexander Svae Sletten

Master of Science in Mechanical Engineering
Submission date: June 2013
Supervisor: Olav Bolland, EPT
Co-supervisor: Lars Olof Nord, EPT

Norwegian University of Science and Technology
Department of Energy and Process Engineering

NTNU

Norwegian University
of Science and Technology

Department of Energy
and Process Engineering

EPT-M-2013-107

MASTER THESIS

for

Student Alexander Sletten

Spring 2013

Optimization of combined cycles for offshore oil and gas installations

Optimalisering av kombinerte kraftprosesser på offshoreinstallasjoner

Background and objective

The power production in the Norwegian offshore oil&gas industry is today generated with gas turbines at low thermal efficiencies. The increasing focus on the greenhouse effect and the introduction of a CO_2 tax have lead to a motivation for reducing the operational cost for power production offshore.

The thesis work is to investigate the use of Combined Cycle technology on oil/gas platforms and productions floaters. The use of a compact heat recovery steam generators (HRSG) with once-through (OT) technology is of particular interest. More specifically, the work includes optimizing the design of the steam cycle to achieve a low weight and volume while having an acceptable performance. For example, the goal could be to minimize the weight-to-power ratio by modifying either the design indirectly (by modifying the process parameters) or by modifying the design directly by changing HRSG tube diameter, fin height, fin spacing, fin thickness, tube layout, material selection, etc. Ideally, the student would first optimize for the process parameters and let the software design the steam cycle. Once the process parameters are optimal, the HRSG geometry could be altered to achieve the best solution for achieving the goals. The second part of the work is depending on software limitations. The work should be done in MATLAB, GT PRO, and Microsoft Excel.

The starting design would be based on the student's project work (Autumn, 2012).

The tasks for the Master thesis include:

1) Selection of optimization objective function(s), design parameters to optimize, and optimization method(s).
2) Optimization by using MATLAB, GT PRO, and Microsoft Excel.
3) Comparisons of results between optimized design and project work design.
4) Comparison of results generated with different optimization methods.
5) Impact of changes in optimization parameters (number of evaluations, lower and upper bounds, etc.).
6) Optimization of HRSG geometry (dependant on possible software limitations).

-- " --

Within 14 days of receiving the written text on the master thesis, the candidate shall submit a research plan for his project to the department.

When the thesis is evaluated, emphasis is put on processing of the results, and that they are presented in tabular and/or graphic form in a clear manner, and that they are analyzed carefully.

The thesis should be formulated as a research report with summary both in English and Norwegian, conclusion, literature references, table of contents etc. During the preparation of the text, the candidate should make an effort to produce a well-structured and easily readable report. In order to ease the evaluation of the thesis, it is important that the cross-references are correct. In the making of the report, strong emphasis should be placed on both a thorough discussion of the results and an orderly presentation.

The candidate is requested to initiate and keep close contact with his/her academic supervisor(s) throughout the working period. The candidate must follow the rules and regulations of NTNU as well as passive directions given by the Department of Energy and Process Engineering.

Risk assessment of the candidate's work shall be carried out according to the department's procedures. The risk assessment must be documented and included as part of the final report. Events related to the candidate's work adversely affecting the health, safety or security, must be documented and included as part of the final report.

Pursuant to "Regulations concerning the supplementary provisions to the technology study program/Master of Science" at NTNU §20, the Department reserves the permission to utilize all the results and data for teaching and research purposes as well as in future publications.

The final report is to be submitted digitally in DAIM. An executive summary of the thesis including title, student's name, supervisor's name, year, department name, and NTNU's logo and name, shall be submitted to the department as a separate pdf file. Based on an agreement with the supervisor, the final report and other material and documents may be given to the supervisor in digital format.

☐ Work to be done in lab (Water power lab, Fluids engineering lab, Thermal engineering lab)

☐ Field work

Department of Energy and Process Engineering, 16. January 2013

Lars E Bakken
Coordinator

Olav Bolland
Supervisor

Co-supervisor: Dr. Lars Olof Nord, NTNU

Preface

This report is written in conjunction with the master thesis Optimization of combined cycles for offshore oil and gas installations. It was written spring 2013 at NTNU, Department for Energy and Process Engineering.

I would like to thank my two supervisors Olav Bolland and Lars O. Nord for the guidance they provided.

Trondheim, June 9, 2013

Alexander Svae Sletten

Alexander Svae Sletten

Abstract

With the increasing focus on the greenhouse effect and the introduction of taxation on NO_X and CO_2 emissions there has been an increased interest in reducing the emissions from the offshore oil and gas installations to bring down the operating costs. This makes the possible use of combined cycles as a source for power production offshore of great immediate interest. Compared to the simple cycle gas turbines typically used offshore today, combined cycles offer a significantly improved thermal efficiency and as such reduced emissions and fuel consumption. However, the large weight and area requirements for combined cycles are a concern; for offshore applications a compact system is needed.

This thesis is an extension of the project work *Process simulation of combined cycles for offshore applications*, written autumn 2012, and focuses on optimizing the design developed in that work. The design parameters for the system developed in the project work were optimized in MATLAB using a connection between MATLAB and a Microsoft Excel spread sheet linked with GT PRO. The thesis includes the development of an objective function and a screening of the potential MATLAB optimization methods. After the optimization methods were decided upon, adjustments were made to them in an attempt to improve the optimized solution, and a brief comparison of the different optimization methods was carried out. Finally, the best solution was compared to that of the project work, both in respect to the individual design parameters and total system performance.

Through this thesis it has become apparent that the selection of objective function is of paramount importance, the optimized solution will only be as good as the selected function. In terms of the optimization methods, there were fairly small differences between the various algorithms, though the pattern search with a MADSPositiveBasis2N search algorithm seemed to be a good option for obtaining the best possible solution. In comparison to the design developed in the project work, there were noticeable improvements to be had in terms of power production and weight savings. Overall, the main components of the optimized solution were 493 kg lighter and able to produce an additional 268 kW when compared to the project work, corresponding to a 2.6 % improvement in the value of the selected objective function. This may not sound like much, but the cumulative savings over the lifetime of an installation may become quite significant. Overall it appears to be quite advantageous to optimize the design of combined cycles for offshore oil and gas installations. Once a suitable objective function is established a quite good optimized solution can be realized in relatively short time. It does not appear to be necessary with many adjustments to the optimization parameters, though adjustments can be made if a better solution is sought after.

Sammendrag

Med økt fokus på drivhuseffekten og innførelsen av CO_2- og NO_x- avgift på utslipp har det vært økende interesse for å redusere utslippene fra offshore olje- og gassinstallasjoner for å få ned driftskostnadene. Dette har gjort den mulige bruken av kombinerte kraftprosesser som en kilde til kraftproduksjon høyaktuell. Sammenlignet med enkle gassturbiner som stort sett brukes offshore i dag tilbyr kombinerte kraftverk betydelig forbedret termisk virkningsgrad og dermed også reduserte utslipp og drivstofforbruk. Den høye vekten og arealbehovet for kombinerte kraftprosesser er imidlertid en bekymring, offshoreinstallasjoner er avhengig av et kompakt system.

Denne avhandlingen er en videreføring av prosjektoppgaven *Process simulation of combined cycles for offshore applications*, skrevet høsten 2012, og fokuserer på å optimalisere det kombinerte kraftprosessanlegget utviklet i den prosjektoppgaven. Designparameterne for det utviklede systemet ble optimalisert i MATLAB ved hjelp av en forbindelse mellom MATLAB og et Microsoft Excel regneark knyttet til GT PRO. Avhandlingen omfatter utviklingen av en objektiv funksjon og screening av potensielle optimaliseringsmetoder i MATLAB. Etter at valg av optimaliseringsmetoder var foretatt ble de justert i et forsøkt på å forbedre den optimaliserte løsningen, og en sammenligning av de ulike optimaliseringsmetodene ble gjennomført. Til slutt ble den beste løsningen sammenlignet med løsningen fra prosjektoppgaven, både med tanke på de enkelte designparameterne og med tanke på den totale systemytelsen.

Gjennom denne avhandlingen har det blitt klart at valg av objektiv funksjon er svært avgjørende for den oppnådde løsningen, og løsningen vil kun bli så god som valget av objektiv funksjon. Når det gjelder optimaliseringsmetodene var det ganske små forskjeller mellom de ulike algoritmene, men pattern search med en MADSPositiveBasis2N søkealgoritme virket å være et godt alternativ når det gjaldt å finne best mulig løsning. I forhold til designet utviklet i prosjektoppgaven var det merkbare forbedringer å finne både med tanke på kraftproduksjon og vektbesparelse. Totalt sett var den optimaliserte løsningen 493 kg lettere og i stand til å produsere ytterligere 268 kW i forhold til løsningen fra prosjektoppgaven, tilsvarende en 2.6 % forbedring i verdien av den valgte objektiv funksjonen. Det høres kanskje ikke mye ut, men de kumulative besparelsene over levetiden til installasjonen kan bli ganske betydelig. Generelt virker det å være ganske fordelaktig å optimalisere designet av kombinerte kraftprosesser for offshore olje- og gassinstallasjoner. Når en passende objektiv funksjon er etablert er det mulig å oppnå en ganske god løsning i løpet av relativt kort tid. Det virker ikke å være nødvendig med mange justeringer av optimaliseringsparameterne, men dersom en bedre løsning søkes er det fullt mulig å gjøre justeringer.

Acronyms and Abbreviations

CC	Combined cycle
DPHRSG	HRSG draft loss
F-count	Number of function evaluations
GA	Genetic algorithm
GPS	Generalized pattern search
GSS	Generating set search
GT	Gas turbine
HRSG	Heat recovery steam generator
Iter	Number of iterations
LB	Lower boundary
LSP	Live steam pressure
LST	Live steam temperature
MADS	Mesh adaptive search
Pcond	Condenser pressure
PPTD	Pinch point temperature difference
PS	Pattern search
ST	Steam turbine
UB	Upper boundary

Nomenclature

BW	Weight of structures and components needed for the combined cycle offshore in addition to the main components	[kg]
C_i	Any constant	[-]
D	Degree of difficulty	[-]
G_i	Any equality constraint	[-]
H_i	Any inequality constraint	[-]
N	The number of independent variables	[-]
P_{ST}	Power production from the ST	[kW]
T	Total number of polynomial terms in the objective and constraints function	[-]
U	Objective function to be optimized	[-]
U_{opt}	Optimal value of the objective function	[-]
W_{MC}	Weight of the main components in the steam cycle of the combined cycle	[kg]
x_1 to x_n	Independent design variables	[-]
ΔC_i	Slack variable	[-]
ΔP_{GT}	Power production from the GT minus a reference power production from a GT run as a simple cycle	[kW]

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

For offshore oil and gas installations, power generation and mechanical drive typically comes from simple cycle gas turbines. They have a high power to weight ratio, good reliability and can make use of fuel usually readily available at the site. However, they also have a relatively poor thermal efficiency and consequently relatively large flue gas emissions. With the increasing focus on the greenhouse effect and the introduction of taxation on NO_X and CO_2 emissions, this has resulted in increasing operating costs and brought to light a motivation to find power generation alternatives with lower emissions.

The Norwegian government introduced a tax on carbon emissions in 1991. The current offshore CO_2 tax rate for natural gas is at 410 NOK/ton, up from 226 NOK/ton last year. Similarly, a NO_x tax on among others turbines with a total installed capacity above 10 MW and flares on offshore installations was introduced in 2007. It is currently sitting at 17.01 NOK/kg, up from 16.69 NOK/kg last year. In its 2013 government budget, Norway have budgeted for 3 400 000 000 NOK in CO_2 taxes from the continental shelf, while they expect to fetch about 130 000 000 NOK in NO_x taxes. As the budget makes no mention of what sector the NO_x taxes are from, it presumably includes all sectors. However, with petroleum activities' accounting for 29.2 % of the total NO_x emissions in Norway in 2011 it still becomes a sizeable number. With tax costs like that it is quite apparent there are large financial incentives to reducing the emissions from the offshore power generation (Finansdepartementet, 2012; Norwegian Petroleum Directorate, 2013; Tollvesenet, 2013).

Realistic alternatives for reducing the emissions offshore includes increasing the thermal efficiency, cleaning of the exhaust gas or receiving electricity from environmental friendly onshore sources. Of those alternatives, increasing the plant efficiency is a very attractive alternative as it reduces both the fuel consumption as well as reducing the flue gas emissions. In some cases using combined cycles has proven economically preferable to simple cycles in Life Cycle evaluations. Kloster (1999) describes three such solutions, for the platforms Eldfisk, Oseberg and Snorre B.

Kloster (1999) estimates the increased thermal efficiency with combined cycles reduces NO_X and CO_2 emissions by a minimum of 25 % when compared to a simple cycle gas turbine with similar power output. Combined cycles are also known to have very good part-load performance, particularly if multiple gas turbines are connected to a single heat recovery steam generator. Part-load performance is of great importance in offshore applications as operating conditions and requirements rarely matches that of the design point.

However, using combined cycles offshore does have its drawbacks. Particularly the weight and area requirements are a concern, both of which are at a premium offshore and have

large costs connected to them. As such, offshore combined cycles have to be very compact compared to their onshore counterparts (Kloster, 1999).

Nord and Bolland (2012) considered the possible use of a once-through steam generator, as such avoiding the steam drum and possibly bypass stack and thereby saving weight and area footprint. Using the Thermoflow (2012a) product GT PRO such a plant was developed in my project work during autumn 2012, and when compared to a more conventional onshore combined cycle it showed considerable weight savings without affecting the thermal efficiency too much.

1.2 Risk Assessment

In this thesis no laboratory work or excursions has been performed. Consequently no risk assessment regarding this work has been carried out.

1.3 Scope of Work

This master thesis is an extension of my project work *Process simulation of combined cycles for offshore applications*, written autumn 2012. Theory concerning combined cycles (CC) and thermodynamics is handled in the project work, and is as such not repeated here. The plant design originally developed in the project work was thought to have some room for improvements. As such, this thesis focuses on optimizing the design developed in the project work and comparing the results to that of the project work. The work includes a review of general optimization theory, as well as possible optimization methods using the global optimization toolbox available with MATLAB.

The main focus of the thesis is the optimization of combined cycles for offshore oil and gas installations. The plant developed in the project work was used as a starting point, and then the design parameters for that design were optimized in MATLAB by extracting point data from and inserting to a Microsoft Excel spread sheet linked with GT PRO. The thesis includes the development of an objective function and the design parameters to optimize. Then potential optimization methods were assessed by screening the various methods, and considering which methods achieved the best solution as well as how well quickly they did so. After the optimization methods were decided upon they were modified in an attempt to improve the solution of the optimization problem, and a brief comparison of the different methods was carried out. The modifications were mainly changes that essentially affected the number of function evaluations, though other changes in optimization parameters were also considered. Then the best optimized solution was compared to that of the project work, both in respect to the individual design parameters and total system performance.

If the software had easily permitted it, it would have been preferable to also optimize the HRSG geometry.

2 Optimization

In simple terms optimization is the process of finding a solution that gives the maximum or minimum values of a function or quantity. In engineering optimization is typically used to find the best design of a system according to the objective of the design, for instance minimizing the cost for a given output, or maximizing energy efficiency. As such, the quantity or function to be optimized represents the features or aspects of the design that are of particular interest. This quantity or function is known as the objective function. It is a function of the independent variables in the problem, and may be expressed as:

$$U = U(x_1, x_2, x_3, \dots, x_n) \rightarrow U_{opt} \tag{2.1}$$

Where U denotes the objective function to be optimized, x_1 to x_n denotes the n independent design variables in the problem and U_{opt} denotes the optimal value of the objective function (Jaluria, 2008, pp. 429-433; Stoecker, 1980, pp. 131-132).

What a suitable objective function may be depends on the application and type of system. Thermal systems are frequently optimized with respect to aspects like weight, size, rate of energy consumption, efficiency, overall profits, incurred costs or output delivered. If the total cost is to be minimized, the objective function could be the summation of the cost of all components in the system. The optimization can be within a given concept or between different concepts. A complete optimization procedure thus consists of finding all reasonable concepts, then optimizing the design of each concept and finally choosing the best optimized design. The objective function must represent all important characteristics and concerns of the system and its intended application. At the same time, it should be sensitive to variation in design parameters so that a clear optimum can emerge from the optimization process (Jaluria, 2008, pp. 429-433; Stoecker, 1980, pp. 131-132).

In addition to the objective function, the independent design variables are a crucial part of defining the basic problem for the optimization. The number of independent design variables dictates the complexity of the problem, so to limit the complexity it is important to focus on the dominant design variables. Each new independent variable increases the workload attended with solving the problem at hand substantially, particularly for complicated nonlinear problems that are typically associated with thermal systems. As such, optimization is generally carried out using only the most important design variables for the system under consideration (Jaluria, 2008, pp. 438-439).

During the optimization process, parameters are free to float until a combination that optimizes the design is reached. However, there may be some minimum requirements or constraints which limit certain parameters or system designs. As such there is a workable domain that satisfies the constraints and requirements. As can be seen in Figure 2.1 local extrema may be present at different points in the domain, but in most applications there is only one global optimal point. For optimization purposes the interest is in finding the global optimum, but local optima can easily confuse the search for the true global optimum, and

thus complicating the optimization process (Jaluria, 2008, pp. 433-434; Stoecker, 1980, pp. 131-132).

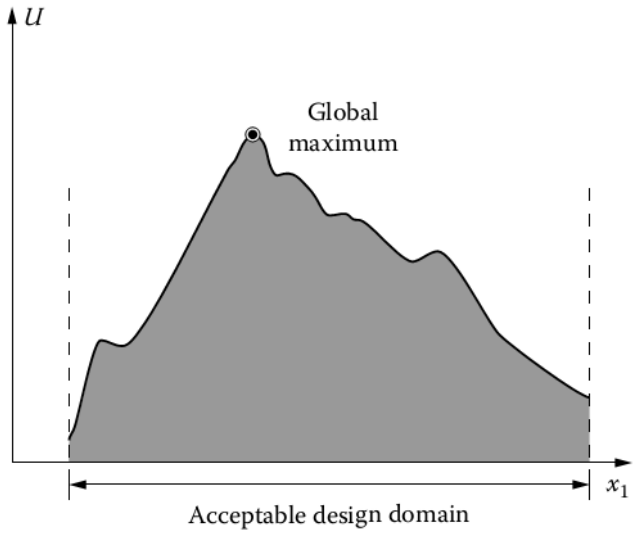

Figure 2.1 – Global maximum of objective function *U* within the acceptable design domain of design variable x_1 (Jaluria, 2008, p. 434).

The constraints for a given problem typically arise from the conservations principles for energy, mass and momentum, or limitations on the ranges of the physical variables. However, the constraints from the conservation principles are often satisfied by the governing equations, as they are usually based on the conservation laws. On such occasions, it is already included in the objective function. If the conservation principles are satisfied, only the additional limitations that define the boundaries of the design domain, like maximum allowable pressure or temperature, remains to be considered (Jaluria, 2008, pp. 434-435).

There are two different types of constraints, equality and inequality. As the names suggest, an equality constraint may be written as an equality function (Jaluria, 2008, p. 436):

$$G_i(x_1, x_2, x_3, \dots, x_n) = 0 \tag{2.2}$$

Where G_i is any equality constraint. An inequality constraint indicates the upper or lower limit of a function, and may consequently be written as an inequality function (Jaluria, 2008, p. 436):

$$H_i(x_1, x_2, x_3, \dots, x_n) \leq C_i \tag{2.3}$$

Where H_i is any inequality constraint while C_i could be any constant. The equality constraints typically arise from the conservation principles, and are as such often already included in the objective function. However, inequalities usually demands more attention, and are typically difficult to solve. Equations and systems of equations are generally easier to deal with due to a greater amount of methods for solving them. Accordingly, inequalities are often converted

into equations by use of slack variables before applying optimization methods. The slack variables are added to the inequality constraints, transforming them into equations, and restricting the design variables to remain within the permitted design domain. A transformation of the inequality constraint from Equation 2.3 could look like this (Jaluria, 2008, p. 437):

$$H_i(x_1, x_2, x_3, \dots, x_n) = C_i - \Delta C_i \tag{2.4}$$

Where the slack variable ΔC_i is a chosen quantity that makes the equation satisfy the original inequality constraint. An optimization problem can also be unconstrained, in which case it is much easier to solve than a constrained problem (Jaluria, 2008, pp. 436-438).

As such, the total optimization problem may be written mathematically as combination of Equation 2.1 through Equation 2.4 (Jaluria, 2008, pp. 436-439):

$$U = U(x_1, x_2, x_3, \dots, x_n) \rightarrow U_{opt}$$

with $$G_i(x_1, x_2, x_3, \dots, x_n) = 0, for\ i = 1,2,3, \dots, m$$

and $$H_i(x_1, x_2, x_3, \dots, x_n) \leq C_i\ or\ \geq C_i, for\ i = 1,2,3, \dots, l \tag{2.5}$$

With *m* needing to be smaller than *n* for there to be an optimization problem. If *m* is equal to *n* there is no need for an optimization as the problem can simply be solved be solving the constraint equations. While *m* being larger than *n* would imply that the problem is over-constrained, and no solution is possible without removing some of the constraints.

When the basic problem is developed, including deciding on design variables, objective function and potential constraints, there are several different optimization methods that could potentially be employed to solve the problem. Most commonly used are calculus methods, search methods and dynamic, linear and geometric programming, where *programming* simply refers to optimization (Stoecker, 1980, pp. 132-136).

Calculus methods are based on the maximum or minimum of a function occurring when the slope of the function is zero. As such, calculus methods for optimization boil down to finding the derivatives of the objective function and the accompanying constraints. Consequently, the equations expressing the problem at hand need to be differentiable in the design domain, and any constraints must be equality constraints, which limits the application of calculus methods (Jaluria, 2008, pp. 440-441; Stoecker, 1980, pp. 146-147).

Dynamic programming is used to obtain the best path of continuous processes, e.g., the best route of a pipeline. As such, it results in an optimal function rather than an optimal state point, which is quite different from most other optimization techniques. It is applicable to staged processes and continuous functions that can be approximated by staged processes, and as such it is frequently used for systems with discrete stages such as a series of compressors, pumps or heat exchangers. With the total path divided in stages, dynamic programming arrives at an overall optimal path by first establishing optimal paths for the

subsections of the problem, and then using the optimum from each subsection in the overall path (Jaluria, 2008, p. 559, p. 588; Stoecker, 1980, pp. 199-201).

Linear programming can be applied when the objective function and the constraints can be expressed as linear combinations of the independent variables. It is an optimization method that can handle large systems with thousands of variables, and as opposed to calculus methods it can also handle inequality constraints. For linear systems, the workable domain takes the shape of a convex polyhedron, illustrated in Figure 2.2, and any optimum occurs somewhere at the boundary of the workable domain. Consequently, a very simple explanation of most linear programming would be that it moves from one corner in the feasible region to whichever adjacent corner shows the most improvement in the objective function. When there are no corners improving the objective function any further, the optimum is reached (Jaluria, 2008, pp. 442-443, pp. 579-581; Stoecker, 1980, pp. 242-244, p. 255).

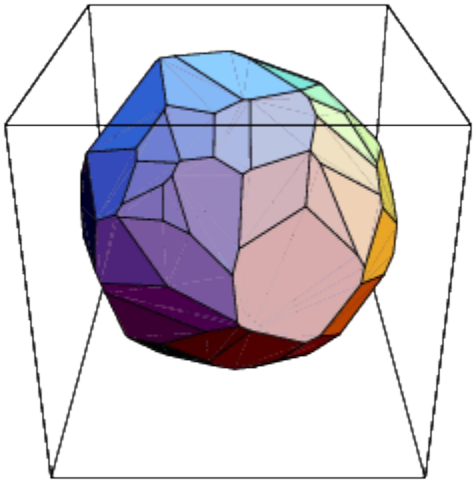

Figure 2.2 – Illustration of convex polyhedron (MathWorld, 2013).

Geometric programming is used to optimize problems where the objective function and the constraints consist of sums of polymers. It differs from most other optimization methods in that it first finds the optimum value of the function rather than values of the independent variables. Additionally, it provides insight into the solution by supplying information regarding the impact each of the design variables have on the solution. For suitable problems it is easy to apply, as it involves solutions of linear equations. However, certain conditions have to be met to make a problem suitable. To determine how suitable a problem is for geometric programming, a parameter called degree of difficulty is used. The degree of difficulty, *D*, is defined as (Jaluria, 2008, p.561; Stoecker, 1980, p. 224):

$$D = T - (N + 1) \tag{2.6}$$

Where *T* is the total number of polynomial terms in the objective function and constraints and *N* is the number of independent variables. When the degree of difficulty is zero it is considered a perfect match for geometric programming, and the problem becomes easily

solvable. When the degree of difficulty is larger than zero the problem is solvable by geometric programming, but will include nonlinear equations and hence become much more time consuming, and other optimization methods are likely to be more efficient (Jaluria, 2008, pp.559-561; Stoecker, 1980, pp. 223-225).

Search methods are based on there being a finite number of designs satisfying the requirements and constraints, and then selecting the best design from the set of acceptable designs. As such, what is needed for a problem to be solvable by search methods are numerical values of the objective function at various locations in the design domain. This makes search methods very versatile, and a large number of methods are developed to handle different problems. Search methods are particularly efficient when discrete points are evaluated, but can also be applied to continuous functions. However, for continuous functions the exact optimum can only be approached, due to the finite number of iterations associated with search methods. In its simplest form, the objective function is calculated at uniformly spaced locations in the acceptable domain, and the optimum value is chosen. However, search methods like that require a large number of simulations, which can be very time-consuming. Consequently, systematic searches that minimize the number of simulations are needed in order to increase the efficiency of the searches (Jaluria, 2008, pp. 441-442, pp. 511-512; Stoecker, 1980, p. 135, p. 169).

Typically there will be regions in the design domain where the objective function approaches its optimal value, and as such it makes sense to focus on those areas. There are two main categories for such search methods; elimination and hill-climbing techniques. As its name suggests, the elimination techniques intends to reduce the domain where the optimum lies by gradually eliminating regions of little interest. An example of this can be seen in Figure 2.3 where the region to the left of A and to the right of C are eliminated as the optimal solution is assumed to be located somewhere between A and C due to the high objective function value at location B.

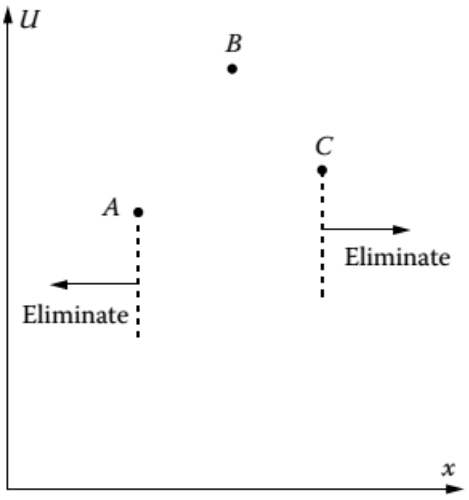

Figure 2.3 – Elimination of regions in search of a maximum U (Jaluria, 2008, p. 517).

Hill-climbing techniques on the other hand use the derivatives of the objective function to gradually move towards the optimum value for the objective function. Figure 2.4 illustrates a hill climbing technique with a lattice search for two variables. Here the search evaluates all points in the lattice immediately surrounding the current point, and moves to the point with the highest value of the objective function. Consequently the search gradually moves towards the optimal solution located in the innermost contour (Jaluria, 2008, pp. 551-553; Stoecker, 1980, pp. 170-180).

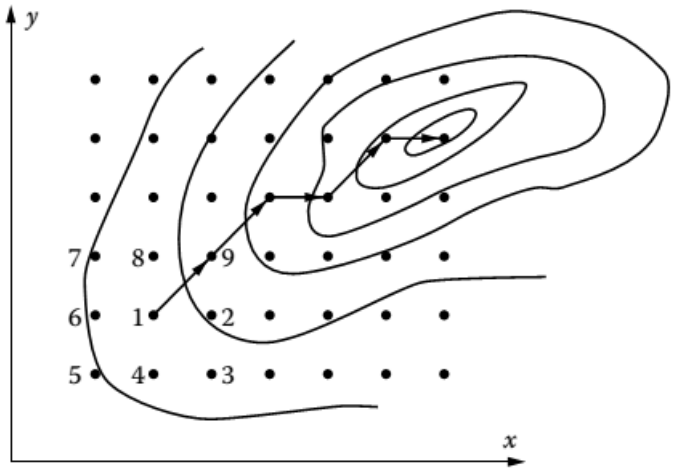

Figure 2.4 – Hill climbing lattice search with two design variables (Jaluria, 2008, p. 528).

2.1 MATLAB Search Methods

This thesis covers optimization of an offshore combined cycle using Thermoflow and MATLAB. Thermoflow calculates the output variables based on the provided input variables. All underlying equations and calculations are hidden, and as such this is a black box problem that is unsuitable for any gradient based optimization methods. Additionally, it is safe to say there are non-linear equations involved, as well as a strong possibility of several local maxima or minima and a possibility of discontinuity. As such, it appears as if search methods are the only viable method for optimizing the problem at hand.

MATLAB comes with a set of built-in search methods in its global optimization toolbox. With the objective function being non-smooth, and the desired solution being the global minimum of the objective function, MathWorks (2012, pp. 1.19-1.20) names three suitable solvers; pattern search (PS), genetic algorithm (GA) and simulated annealing. Of the three methods, PS is suggested as the initial method of preference, with GA generally being less efficient, though being of preference for problems with integer constraints, while simulated annealing is more or less suggested as a last resort, due to its slow convergence. The solver characteristics listed in Table 2.1 are excerpts from a table for all the solvers in the global optimization toolbox (MathWorks, 2012, pp. 1.23-1.25).

Table 2.1 – Solver Characteristics (MathWorks, 2012, pp. 1.24-1.25).

Solver	Convergence	Characteristics
patternsearch	Proven convergence to local optimum, slower than gradient-based solvers.	Deterministic iterates
		Can run in parallel
		No gradients
		User-supplied start point
ga	No convergence proof	Stochastic iterates
		Can run in parallel
		Population-based
		No gradients
		Allows integer constraints
		Automatic start population, or user-supplied population, or combination of both
simulannealbnd	Proven to converge to global optimum for bounded problems with very slow cooling schedule.	Stochastic iterates
		No gradients
		User-supplied start point
		Only bound constraints

2.1.1 Pattern Search

A simple PS is essentially a lattice search. It evaluates the objective function at the current point, creates a mesh around the point and then computes the objective function at the corresponding mesh points. If an improved value of the objective function is found, that point is chosen as the next current point. If no improvement is found, the current point is retained for the next poll. After a successful poll where an improved objective function is found the size of the mesh is increased, while a failed poll results in a reduced mesh size. The pattern of the mesh can be created in various ways. The Global Optimization Toolbox comes with three direct search algorithms for creating the mesh of the pattern search. Generalized pattern search (GPS), generating set search (GSS) and the mesh adaptive search (MADS). The GPS algorithm uses fixed direction vectors, while the MADS algorithm uses a more random selection of vectors. GSS is identical to the GPS algorithm except when there are linear constraints and the current point is close to the constraint boundary. The default pattern for the polling is the GPS Positive basis 2N, consisting of a set of direction vectors. 2N refers to each of the independent design variables having a fixed positive and a negative direction vector. As such, for an objective function with three independent design variables, the GPS Positive basis 2N pattern will consist of the six vectors:

[1 0 0]
[0 1 0]
[0 0 1]
[-1 0 0]
[0 -1 0]
[0 0 -1].

An alternative pattern is the GPS Positive basis Np1, where Np1 refers to N+1 direction vectors. That is one positive unit vector for each independent variable, and then one vector for all the negative directions, giving an objective function with three independent design variables a pattern consisting of four vectors:

[1 0 0]
[0 1 0]
[0 0 1]
[-1 -1 -1].

Consequently an Np1 poll method performs fewer computations per poll, and as such is expected to be quicker. However, a 2N pattern will explore each point more thoroughly and should therefore be better at avoiding ending up in local minima (MathWorks, 2012, p. 4.2, p.4.10, p.4.12, pp. 4.49-4.51).

With PS it is also possible to use a direct search algorithm in addition to the poll. The search algorithm then runs prior to the poll, and attempts to find a point with a lower value of the objective function than the current point. If an improved point is found, that point becomes the current point and no poll is performed at that iteration. If the search is unsuccessful, a poll is performed. The flowchart in Figure 2.5 illustrates the PS algorithm when combined with a direct search method.

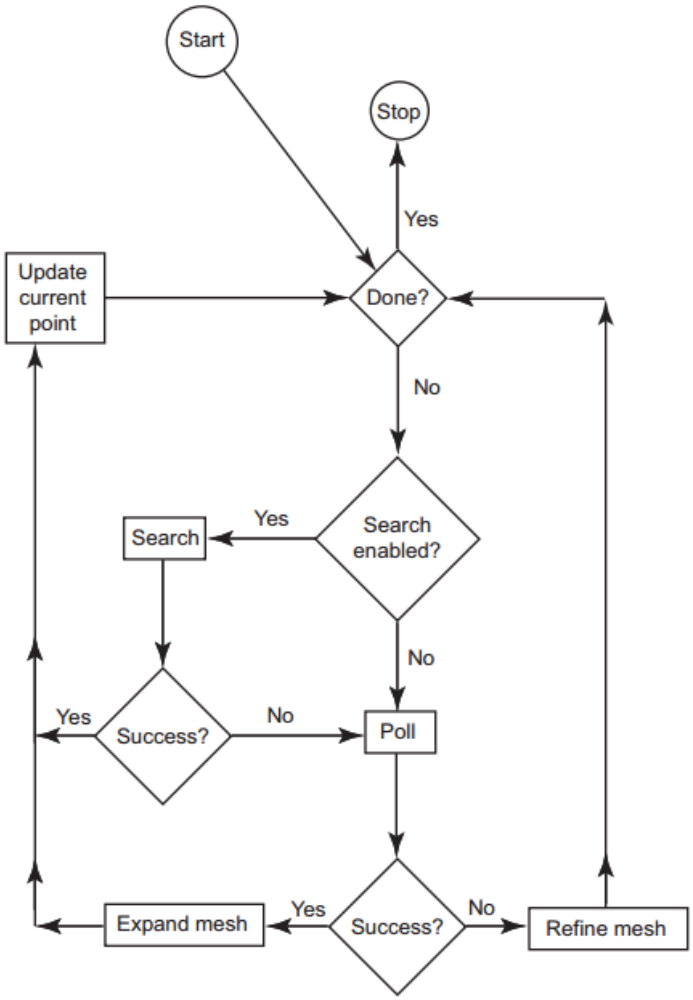

Figure 2.5 – Flowchart of pattern search with a search algorithm (MathWorks, 2012, p. 4.24).

At each iteration step, any of the poll methods can be used as a search algorithm. However, there is no point in using the same algorithm for the search as for the poll. For instance, a GPS poll should be combined with a MADS search, or vice versa. Another option is to use a search method that runs to its completion at the first iteration, and then proceeds with a regular PS after the initial search is completed. Such searches are essentially equivalent to choosing a better starting point for the optimization and possible methods includes GA, Nelder-Mead and Latin hypercube search. GA and Latin hypercube search are both stochastic, while Nelder-Mead only works for unconstrained problems (MathWorks, 2012, pp. 4.23-4.27).

By creating an options structure there are adjustments that can be made to the optimization methods in an attempt to improve their performance, whether that performance gain is in terms of speed of the search or accuracy of the global optimal solution found. For PS the most obvious adjustment possibilities are changing the expansion factor of the mesh following successful polls or the contraction factor of the mesh following unsuccessful polls, changing the stopping criteria for the solver**,** the choice of search method, initial values of design parameters or if a poll should be complete. By complete poll it is meant that all points

in the mesh are evaluated before a potentially new current point is selected. If complete poll is turned off PS will accept the first mesh point that improves the objective function as the new current point. Any remaining mesh points in the current poll will not be evaluated. For problems with several local minima it is sometimes preferable to use complete poll. Additionally, vectorizing the objective and constraint functions will generally lead to an increase in speed as all points in a poll or search pattern can then be evaluated at the same time (MathWorks, 2012, p. 4.51, pp. 4.65-4.73, pp. 4.80-4.81, p. 4.84).

PS has a number of stopping criteria that determines when the algorithm should stop. *Mesh tolerance* specifies how small the mesh size can become before the algorithm halts. *Time limit* puts into place a limit on the allotted time before stopping, while m*ax iteration* and m*ax function evaluations* determines how many iterations and function evaluations the algorithm can potentially carry out before terminating. Additionally there are some tolerance limits. *X tolerance* specifies the minimum distance between the current points at consecutive iterations, and the algorithm terminates if the distance, in conjunction with the mesh size, is smaller than the specified limit. *Function tolerance* is similar, only with the limit being on the objective function. If the change in function value and size of the mesh after a successful poll is smaller than the specified limit, the algorithm stops. Lastly there is the n*onlinear constraint tolerance* that tests the feasibility of the point with respect to nonlinear constraints, and deems the point feasible if the constraint violation is less than the specified limit (MathWorks, 2012, pp. 4.80-4.81, pp.9.21-9.22).

2.1.2 Genetic Algorithm

The idea behind genetic algorithm is that over generations the best individuals in a population will evolve towards an optimum by natural selection. The fittest individuals in a population have the best chance of surviving and passing on their genes to the next generation. In genetic algorithm the objective function is termed fitness function, and any point where the fitness function can be evaluated is an individual. As such an individual is a vector of the design variables, and generally the lower the values of the fitness function, the fitter the individual is. However, some form of transformation is required as using the fitness function value directly is usually ineffective. The population is an array of individuals, while each successive population is termed a generation (Edgar, Himmelblau and Lasdon, 2001, pp. 400-402; MathWorks, 2012, p. 5.2, pp. 5.16-5.18).

In MATLAB the initial population is created at random, though if desired it has the possibility of adding specific individuals to the population or specifying the range from which the population should be generated. Sequences of new populations are created using the current population to create the next population. Members of the current population are selected as parents based on their fitness value. The selection is random; though the fitter the individual is the more likely it is to be selected as a parent. New individuals for the next population, children, are produced from the parents, either by random changes to a single parent or combining the vectors of a pair of parents. The random changes are called

mutation, while the combination of two parents is called a crossover. When two parents are combined in a crossover; at each coordinate of the child vector an entry at the same coordinate is randomly chosen from one of the parents. For instance, for a problem with three design variables, two parents with respectively [2 4 7] and [3 1 9] vectors might produce a [2 4 9] child. Here *2* and *4* stems from the first parent, while the *9* is passed on from the second parent. For the creation of the mutation children, the algorithm randomly changes the entries of the parent, so that for instance a [2 4 7] parent produces a [3 4 7] child. The best individuals in a population are deemed elite, and are carried over unchanged to the next generation. Together with the produced children they replace the current population to form the next generation (Edgar, Himmelblau and Lasdon, 2001, pp. 401-403; MathWorks, 2012, p. 5.2, pp. 5.16-5.22, pp. 5.88-5.89).

For the GA to be efficient, the initial population needs to cover as much space of the interesting region as possible. That is achieved by having a large average distance between the points, a high diversity. Figure 2.6 shows how the population at the left with a high diversity is able to cover a large region of space, while the low diversity population on the right only covers a small space. If the approximate location of the optimal solution is known or suspected, an initial range of the individuals in the population can be set in MATLAB so that the GA searches that region thoroughly. However, as some of the mutation children will come into being outside the suspected region, the GA can find the optimum even if it lies outside the specified range (MathWorks, 2012, pp. 5.17-5.20, pp. 5.74-5.75).

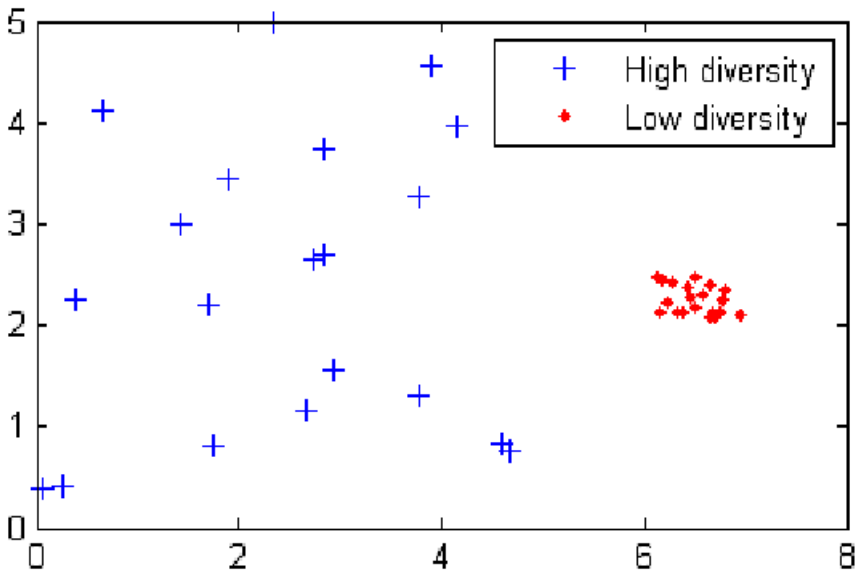

Figure 2.6 - Comparison of high diversity, blue, and low diversity, red (MathWorks, 2012, p. 5.17).

There are a number of stopping conditions for the GA. One can specify a limit on the number of *generations* that should be generated, or a *time limit* on how long the algorithm can run. Other options includes stopping the simulation when the best fitness function reaches a specified limit, or if it fails to improve over a certain number of generations or time,

respectively called *fitness limit*, *stall generations* and *stall time limit*. The last stopping condition for unconstrained problems, *function tolerance*, is slightly more complicated as it stops the algorithm when the weighted average relative change of the value of the fitness function over *stall generations* is less than the specified *function tolerance*. If any of the stopping conditions are met, the algorithm will stop. Additionally there is a stopping condition that is not really used as a stopping criterion, the *nonlinear constraint tolerance*. Its purpose is to test the feasibility of the point in respect to nonlinear constraints (MathWorks, 2012, pp. 5.24-5.25).

There is a long list of possible adjustments for the GA described in the Global Optimization Toolbox User's Guide by MathWorks (2012, pp. 9.31-9.54). In order to increase the accuracy of the solution a larger *population size*, more *generations*, no *time limit*, a decreased *elite count* and a reduced *function tolerance* should be used. By increasing the size of the population the algorithm searches more points, and increases the chance of finding the global minimum, with the default value being 15 times the number of variables, and the most common size being somewhere between 50 to 100 individuals. However, having to search more points naturally causes the algorithm to run slower. Increasing the number of *generations* and having no *time* limit allows the algorithm to continue running until one of the other stopping conditions are met, possibly avoiding stopping the search while there is still room for improvement of the fitness function. A smaller *elite count* increases the search of the design space at each generation as fewer individuals are kept unchanged from one generation to the next. However, with multiple local minima a higher *elite count* might allow the algorithm to explore more minima simultaneously, possibly finding the global minimum quicker. A low *function tolerance* allows the algorithm to continue searching until the elite member of the population changes very little. A good initial guess of where the global optimum is located can help speed up the algorithm, as parts of the initial population then should have a pretty good fitness function value. As mentioned previously an *initial population range* surrounding the suspected region can be set, while if a specific point is known or suspected to be close to the global optimum it can be included as an individual in the *initial population*. Additionally, as with PS, vectorizing the objective and constraint functions will generally lead to an increase in speed as all points in a population can then be evaluated at the same time (Edgar, Himmelblau and Lasdon, 2001, pp. 401; MathWorks, 2012, p. 5.43, p. 5.83, pp. 5.108-5.111, p. 7.5, pp. 9.36-9.37, p. 9.42).

2.1.3 Simulated Annealing

The last potential direct search MATLAB solver is simulated annealing, though as it generally converges much slower than either PS or GA it is more of a backup solution if the other methods fail to converge. The method models the physical process of annealing metals. If a heated metal is cooled rapidly from a molten state irregularities will appear in the crystal structure, and the energy level in the resulting solid will be much higher than in a perfect crystal. When annealing a metal it is cooled very slowly by holding the temperature steady at a series of levels until it reaches thermal equilibrium with the environment. This minimizes

the energy level in the final crystal, and decreases the defects. For the simulated annealing algorithm, the objective function is analogous to the energy level of the metal, and following an annealing schedule where the metal is cooled by systematically lowering an imagined temperature should hence minimize the objective function. As the pretended material cools, while the system is attaining thermal equilibrium at any given temperature, the state changes in a random way, though at lower temperatures it is more likely to transition into states with lower energy than it is at higher temperatures. To achieve this, the algorithm randomly generates a new point at each iteration, with the distance from the new point to the current point depending on a probability distribution with a scale proportional to the temperature. The algorithm accepts all new points that lower the objective function, though with a probability depending on the temperature there is also a chance of a point that raises the objective function value being accepted. The higher the temperature is, the higher the probability of accepting a point that increases the objective function value is, but as the temperature approaches zero the probability of a point that raises the objective function value being accepted approaches zero. This prevents the method from being trapped in a local minimum (Edgar, Himmelblau and Lasdon, 2001, pp. 399-400; MathWorks, 2012, pp. 1.23-1.25, p. 6.2).

Details concerning simulated annealing in MATLAB can be found in the Global Optimization Toolbox User's Guide by MathWorks (2012, pp. 6.2-6.21) but will not be treated here.

3 Methodology

This section briefly describes the simulation tools used in this thesis, and the procedure for the selection of objective function, choice of design variables with upper and lower bounds and the choice of optimization methods.

3.1 Simulation Software

As this thesis is an extension of my project work, the simulation software used and described in my project work is also used in this thesis. The description of said software, the Thermoflow products GT PRO (2013a), GT MASTER (2013b) and PEACE (2013c), are consequently covered in that thesis. Version 22 of Thermoflow is used in this thesis, as it was the most up to date version when the work with the thesis was started.

MATLAB R2011a, from The MathWorks, Inc., is used to program the optimization methods and objective function, and then simulate said optimizations. GT PRO is then linked with Microsoft Excel 2010 and MATLAB, with the help from Lars Nord (2013), and together they are used to compute all the discrete points used in the optimization process. GT MASTER is used to compare the off-design performance of the optimized system with that of the system from my project work, while PEACE is used to estimate the weight of some of the components. The offshore CC plant model used is the one developed in the project work.

3.2 Choice of Objective Function and Design Parameters

The objective function was chosen by considering the motivation behind using a CC offshore and the main drawbacks in doing so. Or in other words, trying to maximize the advantages and minimize the disadvantages associated with installing a CC offshore. There are no apparent constraints associated with the problem at hand, only physical and software limitations handled by upper and lower bounds in design parameters and conservation laws already included in the equations used by the GT PRO software.

For design variables, the variables believed to have the most impact on the objective function was chosen. Upper and lower bounds for the parameters were selected based on what was deemed feasible in regards to previously studied literature and limitations in physical parameters and software.

3.3 Choice of Optimization Methods

The choice of optimization method was done by screening the potential algorithms. The screening of the optimization methods was performed by running a series of optimization methods at similar settings and comparing their resulting value of the objective function and time it took completing or approaching an adequate value. Additionally, the viability of the method, as in how often the simulation crashed, was taken into consideration. It is worth nothing that the setup with GT PRO, Microsoft Excel and MATLAB can only compute one point at a time, making vectorizing and parallel computing of very little interest to this thesis, and as such it was not taken into consideration when selecting optimization methods.

4 Results and Discussion

4.1 Objective Function

Naturally, the first step of the optimization process was to decide upon a suitable objective function. The motivation behind using a CC offshore is to increase the efficiency of the power production, and thereby reducing fuel consumption and CO_2 emissions. This is obtained by extracting additional power from the waste heat from the gas turbine (GT) by adding a heat recovery steam generator (HRSG) and connecting it to a steam turbine (ST). As such, it made sense for the objective function to include the power gain from switching from a simple cycle GT to a CC system. However, an offshore CC also has a considerable drawback in the form of a considerable investment cost associated with its weight and space requirements (Kloster, 1999), which also should be included in the objective function.

With optimization methods in MATLAB generally looking to minimize the objective function, combining the power gain and additional weight in a way where minimizing the objective function would maximize the power gain and minimize the weight seemed necessary. Accordingly, an initial objective function of additional weight per additional power appeared to be quite sensible. The power gain from switching to a CC obviously includes the power from the ST; however, realizing the pressure loss associated with the HRSG has an impact on the performance of the GT, the delta power from the GT would also have to be included in the total power gain. The cost of the additional weight and space is difficult to estimate, the industry has understandably little interest in sharing their data regarding this area. However, discussions with Lars Nord (2013) and Olav Bolland (2013) suggested weight to be dominating the cost picture, and assuming the investment cost to be proportional to the additional weight from the CC system seemed like an adequate assumption for the purpose of this thesis.

In the project work only the weight of the main components was extracted, so the initial thought was to just continue using the weight of the main components. However, with using only the weight of the main components it quickly became apparent that the optimization methods was focusing very heavily on the weight savings, and almost ignoring the power production aspect. After discussions with Nord (2013) it was suggested to add a bulk weight to the weight of the main components, and hence more accurately represent the real situation as well as making the changes in objective function due to weight changes less volatile. Consequently, the principal objective function to minimize ended up as:

$$Objective\ function = \frac{W_{MC} + BW}{P_{ST} + \Delta P_{GT}}$$

With W_{MC} being the weight of the main components in the steam cycle of the CC, BW being the bulk weight corresponding to the weight of structures and components needed for the combined cycle in addition to the main components, while P_{ST} is the power production from the ST and the ΔP_{GT} is the power production from the GT minus a reference power

production from a GT run as a simple cycle with exhaust losses of 4,99 mbar, corresponding to 32504 kW for the GE LM2500+RD (G4) GT used in this thesis. The weight of the main components is using the wet weight the GT, ST, HRSG and the condenser. The bulk weight estimation includes estimations of the bottom frame for the ST and HRSG skid, water- and chemical tanks, motors and pumps, water treatment system and so on, and will be elaborated on later in this thesis.

With an objective function focusing on the weight to power ratio of the additional system associated with a CC, it might be interesting to consider what the approximate weight to power ratio of an offshore simple cycle GT is. According to GT PRO, the GT used in my project work and this thesis produces about 32 MW and weighs about 222 250 kg, while Hellberg (2006) writes that the weight of the skid for Siemens 29 MW SGT-700 GT is about 78 metric tons. By assuming a similar skid structure for the GE LM2500 as well as scaling the size to fit 32 MW, and further assuming a linear relationship with the power production of the GT and the weight of the corresponding GT skid, an estimated weight of the entire GT skid would be around:

$$Weight\ GT\ skid = \ 222\ 250\ kg + \frac{32\ MW}{29\ MW} 78\ 000\ kg = 308\ 250\ kg$$

Kloster (1999) indicates the weight of a GT skid is comparable to the weight of a ST skid with similar power production, and estimates that a ST skid producing between 15 to 20 MW would weight somewhere between 150 to 175 tons. Extrapolating those numbers to fit a 32 MW GT suggests a weight range from:

$$Weight\ range\ GT\ skid = \frac{32\ MW}{20\ MW} 150\ \ tons - \frac{32\ MW}{15\ MW} 175\ tons = 240 - 375\ tons$$

The initial estimation of right above 300 tons fits right in the middle of this range, and seems like a decent estimation. This weight, with a 32 MW GT power production, would correspond to a weight to power ratio of about 10.

For comparison, the estimated weight of the main components in an onshore high efficiency CC developed in my project work, not including the GT, was 444 930 kg, with a gross additional power compared to a simple cycle GT of 13155 kW. Resulting in a weight to power ratio of about 34, and that is without considering any skids needed offshore.

4.1.1 Estimation of Bulk Weight

The following crude weight estimation of the bulk weight for the objective function are based on among others data from GT PRO and PEACE. Discussions with Nord (2013) suggested the weight of the bottom frame for a ST skid producing about 17.3 MW being in the range of 40 tons, while as previously mentioned Hellberg (2006) estimates a GT skid producing about 29 MW weighting about 78 tons. The power production from the ST in the offshore CC developed in my project thesis was about 11.2 MW, and it seems reasonable to expect the power production from the optimized plant to be in the same range.

Extrapolating the figures indicated from the personal communication with Nord suggests a bottom frame weight for the ST of about 26 tons, while extrapolating the numbers from Hellberg (2006) are indicative of a weight around 29 tons. This seems to be in agreement with the suggestion Kloster (1999) made about a ST and GT skid being about the same weight, and using an average of the two as an estimation of the weight of the bottom frame for the ST in the optimized plant seems reasonable. The weight of the HRSG bottom frame is more difficult to estimate, but it seems safe to say it would weigh at least as much as the ST bottom frame, as it in comparison to the ST bottom frame needs additional structure to keep the HRSG in place.

GT PRO suggests three types of water tanks for demineralized, raw and neutralized water, in addition to a water tank for fire protection. A raw water tank seems a bit superfluous offshore, though some form of raw water source, either in the form of transportation from land or desalination plant is needed. It is assumed that some sort of fire protection system is already present at the platform.

In order to establish what sort of water tank sizes GT PRO indicates for different system water levels, the High Efficiency onshore plant developed in my project work was used. The condenser pressure was varied from 0.009 bar to 0.14 bar, resulting in water levels ranging from 21 170 kg to 42 060 kg. Even with these vastly different water levels, the suggested tank sizes never changed much, ranging from 44 810 l to 48 060 l for demineralized water tanks, and from 22 410 l to 24 030 l for neutralized water. In comparison, the suggest tank sizes for the developed offshore OTSG system with water levels of about 8 000 kg were respectively 40 230 l and 20 110 l. As such, it appears as if the suggested tank sizes are only weakly dependent on the water levels in the system. A possible explanation is that the plants developed in GT PRO are intended for onshore use, where the size of the tanks has fairly little impact on the plant costs, and that GT PRO as such operates with a more or less “standard” tank sizes. Based on the suggested tank size for various GT PRO designs, demineralized water a tank size of about 1.5 times the water content in the system, and 0.75 for the neutralized water, seems more than adequate. That would imply the required volume of the water tanks being about 18 000 l for a system containing 8 000 kg of water. Assuming one litre of water weighting 1 kg, and allowing for the tanks self-weight, 20 metric tons might be an adequate estimation of the weight of the water tanks needed for the combined cycle.

The chemical tanks for acidic and alkaline solutions used to clean the water are indicated by PEACE to each hold about 2410 l. Assuming a density of about 1.05 kg/l for the acidic solution and about 1.25 kg/l for the caustic soda, the total weight of the tanks would then slightly exceed 5.5 tons in total (The Engineering ToolBox, 2013).

Using the PEACE results from the offshore CC plant developed in my project work, the total weight of motors and pumps was estimated to about 11 500 kg, with a list of the various units and corresponding weight accessible in Appendix A. The same PEACE data was used to

estimate the weight of the control centre to 5000 kg, and the weight of a step-down transformer also to 5000 kg. It was assumed all the produced power would be used at the platform or in the immediate proximity, rendering a step-up transformer for long distance transportation needless. For the water treatment system PEACE suggest weights between 4000 kg and 36 000 kg depending on the water purity, referred to as total dissolved solids and turbidity. However, discussions with Nord (2013) indicated a value closer to, if not above, 40 000 kg might be more realistic for an offshore CC.

The rounded estimation of all the weights used in the bulk weight estimation can be seen in Table 4.1. The total bulk weight used is rounded up to account for the HRSG bottom frame estimation likely being too conservative, as well as some components, like a separate skid for the feed water pumps to deliver the necessary NPSH, being omitted in this analysis.

Table 4.1 – Weight estimations bulk weight

	Estimated weight [kg]
Bottom frame (ST)	27 500
Bottom frame (HRSG)	27 500
Water tanks	20 000
Chemical tanks	5 500
Motors and pumps	11 500
Control centre	5 000
Transformer (step down)	5 000
Water treatment system	40 000
Total weight	142 000
Total bulk weight, rounded up from the total weight	**150 000**

As such, a very crude estimation of the bulk weight of about 150 000 kg appears to be quite reasonable, and though it contains a lot uncertainties and qualified guesses it should be adequate for this thesis. It should certainly be closer to reality than ignoring the bulk weight altogether.

The MATLAB code for the objective function can be found in Appendix B. It is worth noting that when a GT PRO computation returns a warning message, the objective function value at that point will be set to a very high value so that the point is essentially ignored.

4.2 Design Variables

When it came to choice of design variables to focus the optimization on, it seemed logical to focus on the parameters with the largest influence on the system weight and power production. As in my project work, the parameters believed to have the most impact are the live steam pressure (LSP), the live steam temperature (LST), condenser pressure, pinch point temperature difference (PPTD) and HRSG draft loss.

In order to limit the space of the region the optimization methods would have to search, a lower boundary (LB) and an upper boundary (UB) were used. Based on the results in my project work, the most interesting region in terms of maximum performance and minimum weight appeared to be from 15 to 35 bar as seen in Figure 4.1 For the LST there is a physical upper limitation in that the GT exhaust temperature is about 530 °C, and with the need of a temperature difference to transfer the heat any LST above 510°C is probably unfeasible. Additionally, as also seen in Figure 4.1 the weight savings of reducing the temperature below 455 °C appears negligible, with the only result being lower plant efficiency. Eventually, a lower bound of 400 °C was selected, though it could probably safely have been increased to around 455 °C.

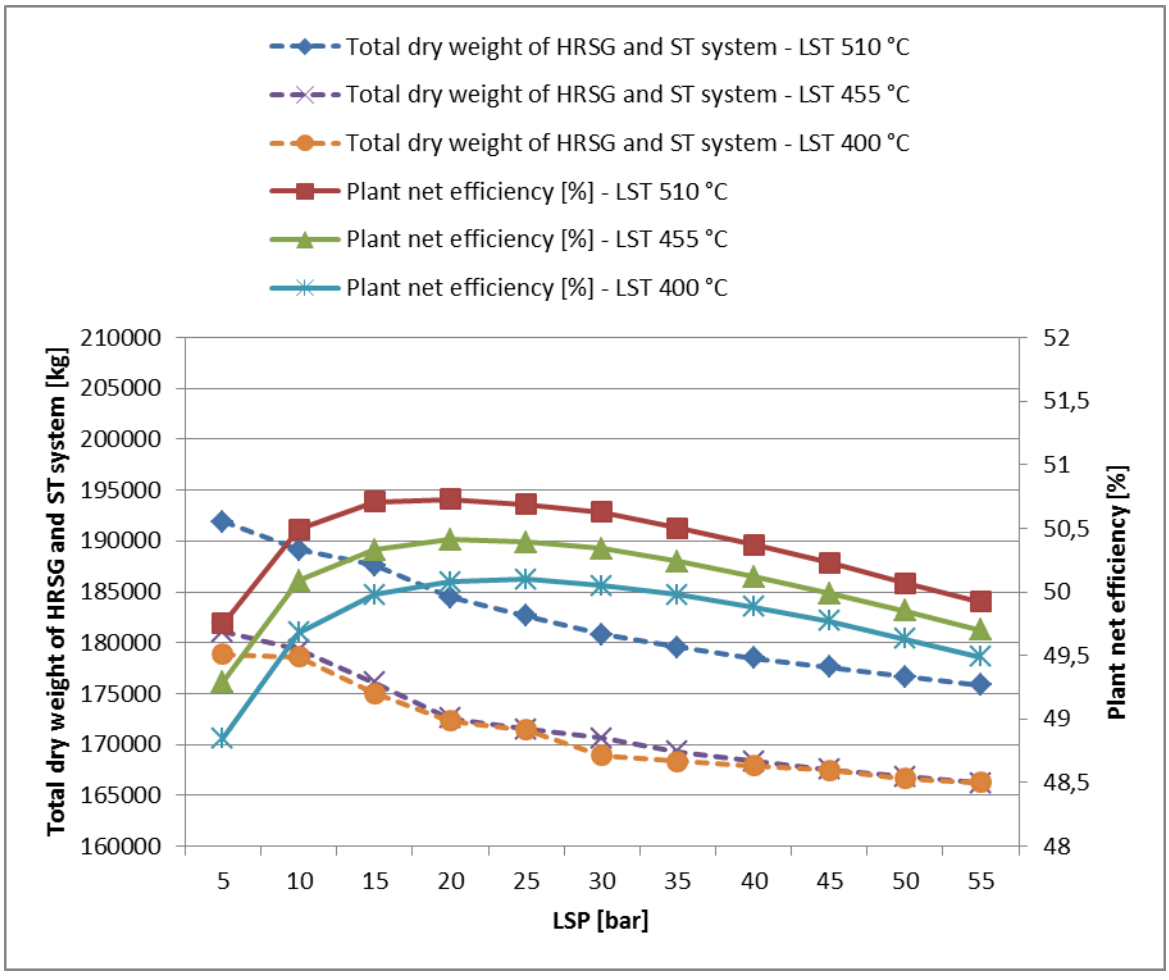

Figure 4.1 - Effect of LSP on weight and plant efficiency for different LSTs (Sletten, 2012, p. 30).

As for the condenser pressure (Pcond), the lowest feasible limit with the selected ambient water temperature is 0.0234 bar, while Figure 4.2 shows that any weight savings past 0.14 bar seems negligible. Considering it is unknown whether the best course of action to optimize the system is to cut weight or increase power production, the entire range from 0.0234 to 0.14 bar was kept. For the PPTD, the entire range from 8 to 35 °C suggested in theory, and shown in Figure 4.3, was also kept, though it is probably unlikely that the lower end of that range will feature in the optimized solution. The same could probably be said for

the HRSG draft loss, where the weight savings pertaining to an increase in draft loss is considerable in comparison to the performance loss. Still, the range from 15 to 35 mbar used in the project work was retained here. For both the PPTD and the HRSG draft loss (DPHRSG) it is quite possible that a better solution might exist outside the suggested upper limit. However, GT PRO struggles when the cross-sectional area of the HRSG gets too small, and fails to compute accurately when the mass flux surpasses 6.103 kg/m^{2}s. With an inaccurate computation GT PRO accompany the calculation with a warning message, and any such point will essentially be ignored in this optimization process. The cross-sectional area of the HRSG typically reduces when the PPTD and HRSG draft loss increases, and simulations with GT PRO in the region above the suggested upper limit for those two parameters proved very difficult, with most computations returning a warning message and occasionally crashing the simulation. So in order to limit crashes and somewhat simplifying the optimization process, the upper limits were kept as is.

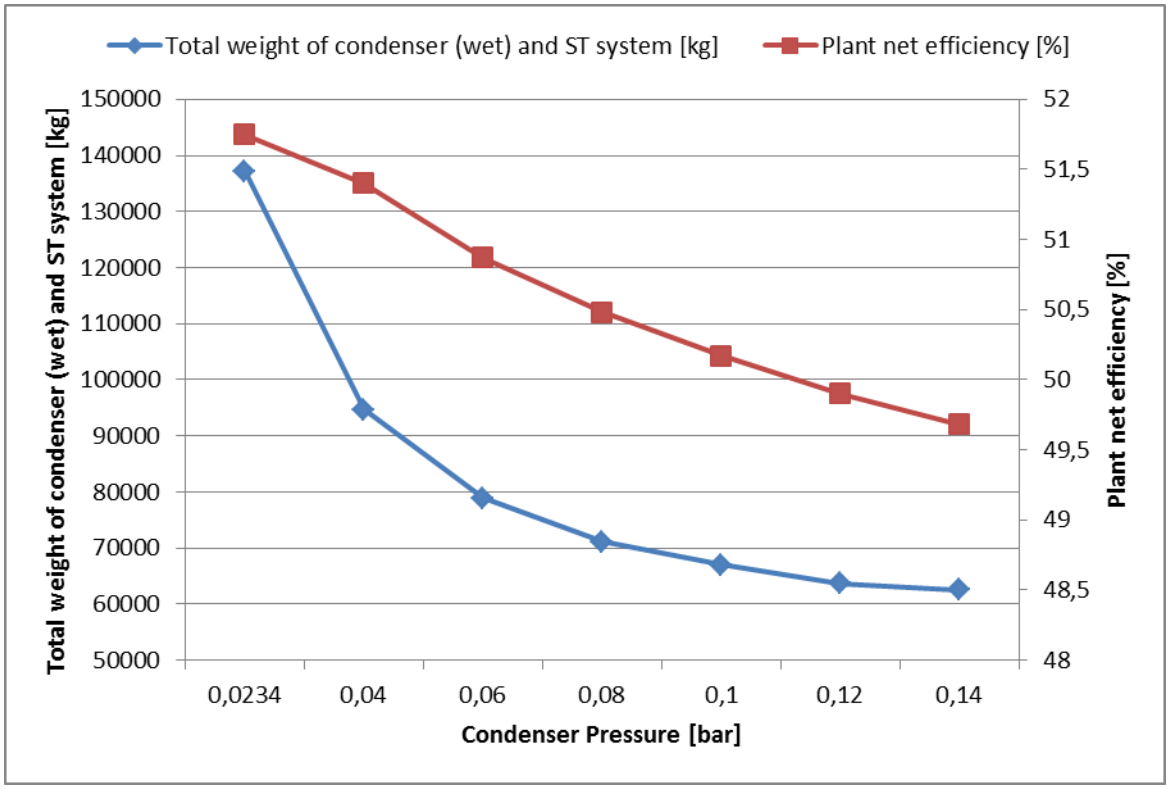

Figure 4.2 - Effect of condenser pressure on weight and plant efficiency (Sletten, 2012, p. 29).

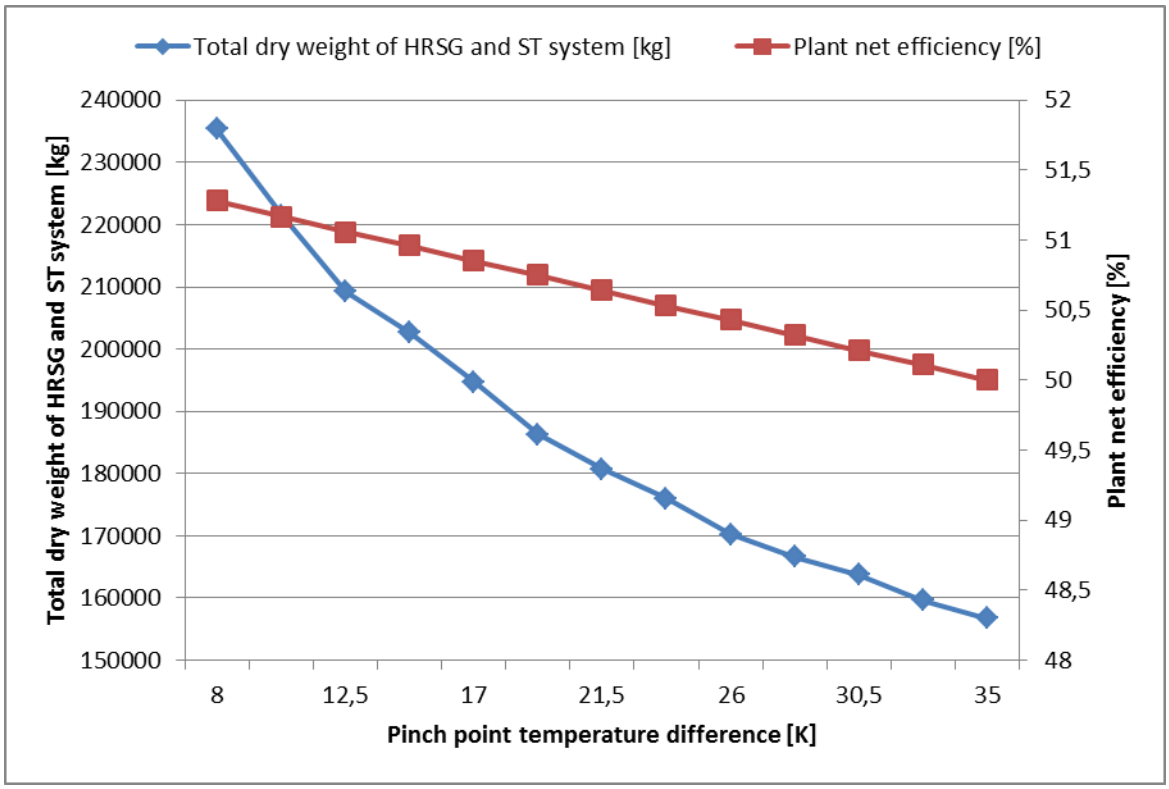

Figure 4.3 - Effect of PPTD on weight and plant efficiency (Sletten, 2012, p. 28).

A summary of the upper and lower boundaries used for the design parameters can be seen in Table 4.2.

Table 4.2 – Upper and lower bounds of design parameters

	Lower boundaries	Upper boundaries
LSP [bar]	15	35
LST [°C]	400	510
Condenser pressure [bar]	0.234	0.14
PPTD [°C]	8	35
HRSG draft loss [mbar]	15	35

4.3 Choice of Optimization Methods

As discussed in the theory, there were three distinctively different built-in search methods in the MATLAB global optimization toolbox that are suitable for the problem at hand. Of those, PS should in theory be the quickest, followed by the GA and lastly the simulated annealing. PS also comes with a large number of possible search algorithms that can be added to the general algorithm, possibly improving its performance, though vastly increasing the number of possible search algorithms. The basic MATLAB codes for the different optimization methods are available in Appendix B.

In order to narrow down the number of possible search algorithms, an initial screening of the methods was performed, with the aim of ending up with 2-3 different optimization

algorithms to move on with. Under the screening process it was considered how quickly an adequate objective function value was achieved, though the main emphasis was placed on the best obtained value of said objective function. With PS expected to both being the quickest and having the most alternatives to investigate, it was the main emphasis of the initial screening process. To validate and put any PS results into perspective it was preferable if one of the 2-3 methods retained for further analysis were fundamentally different, and as such GA and simulated annealing were explored as potentially reference search methods.

As described in the theory, there are two fundamentally different ways to apply additional search algorithms to the PS. There are the GPS, GSS and MADS algorithms that run concurrent with the regular PS, either with a Np1 or 2N patterns, and then there are the GA, Nelder-Mead and Latin hypercube searches that runs prior to the regular PS. Additionally, it is possible to use different polling methods for the regular PS. The initial screening used the default GPSPositiveBasis2N method. Attempts were also made using MADSPositiveBasis2N. However, as it proved impossible to modify the mesh expansion and contraction using that polling method, the polling mesh changes ended up too crude to find any points close to the optimum, and it was consequently discarded as a potential polling method for this thesis. Further, Np1 pattern for the polling was not attempted, as evaluating each point more thoroughly using a 2N pattern was believed to be better considering the objective function was expected to include several local minima.

For the search methods, based on theory one would expect Np1 patterns having fewer function evaluations and as such being quicker in reaching its stopping criteria, while a 2N pattern should be less likely to be trapped in a local minima, and consequently possibly obtaining a better objective function value. Considering GSS searches are supposed to be identical to GPS searches when there are no linear constraints, they should perform very similarly. As the default poll method is GPSPositiveBasis2N, one would expect a different search method, like MADS, to work the best, while a GPSPositiveBasis2N search method should make absolutely no difference from just the regular PS. For the algorithms running in advance of the regular PS it is difficult to predict the performance based on the available theory, the typical course of action seems to be trial and error.

4.3.1 Initial Screening

Initially, a couple of GA simulations were run in order to have a reference point to compare the various PS search algorithms with. Population size was believed to be very important for the result, and with the size typically being somewhere between 50 and 100, it was decided to test both outliers on that region. Simulated annealing was also tested, but with the computer crashing each time it was attempted it was quickly given up on and discarded as a potential search method for this thesis. In any event it was expected to be by far the slowest method, and with the GA performing fine, trying to get the simulated annealing to work did not seem to be worth the time required.

The initial GA simulations applied a 1e-7 function tolerance limit as the main stopping criteria, the stopping criteria expected to terminate the simulation, and included the values of the design variables from the project work in the initial population. The thought process behind including the values from the project work in the GA initial population was that the PS the GA is supposed to be a reference to will need an initial point to start its search from, and as such it would probably be beneficial if the two methods started from the same location, and hence not putting one of the methods at a disadvantage right off the bat. Further, when deciding on what the initial location should be, the design variable values developed in my project were assumed to be relatively close to an optimal solution, and should at the very least provide a decent starting point for the optimization process. Using the design parameters from the project work corresponds to an objective function value of 30.4612. With that as the only tweaks to the default GA algorithm, Table 4.3 shows the differences between the two population size outliers. Note that Iter is short for Iterations, while F-count is the number of function evaluations. The GA with a population size of 100 took roughly 50 % more time to complete than the population with 50 individuals, but ended up with a better objective function value. The GA with a population size of 100 was kept as a reference search method as the value of the objective function is ultimately more important than the time to reach it.

Table 4.3 – GA with population size of 50 and 100 respectively

	Objective function value	Iter	F-count	Time [h]	LSP [bar]	LST [°C]	Pcond [bar]	PPTD [°C]	DPHRSG [mbar]
GA - 50	29,7297	42	2150	02:01:06	24,6	488,7	0,054	33,23	34,98
GA - 100	29,6889	36	3700	03:27:08	22,9	492,2	0,047	34,99	34,99

With the initial location already decided on, the only change to the default options structure for the initial PS screening was setting complete poll to *on*, as it was suspected the problem would have multiple local minima, and it is often preferable using a complete poll in such scenarios.

A straightforward PS was then performed. However, it seemingly got stuck in the same local minima as the GA with a population size of 50. It was thought that a possible reason for this could be that the mesh from the polling would be too coarse with the default expansion and contraction factor of respectively 2 and 1/2, and as a consequence miss the region surrounding the global optimum when the mesh size contracted. To test this hypothesis a second PS was performed, now with an expansion factor of 3 and a contraction factor of 2/3. As can be seen from Table 4.4 this change did in fact make the PS locate a point in the same region as the GA with a population size of 100, and as predicted taking less time in doing so. With basically the same arguments as for selecting the GA with a population size of 100, it was decided to keep the PS with an expansion factor of 3 and a contraction factor of 2/3 for the remaining of the initial screening process.

Table 4.4 – Default pattern search compared to a pattern search with an expansion factor of 3 and a contraction factor of 2/3

	Objective function value	Iter	F-count	Time [h]	LSP [bar]	LST [°C]	Pcond [bar]	PPTD [°C]	DPHRSG [mbar]
PS - default	29,7344	300	2925	02:44:01	23,9	488,6	0,054	33,73	34,52
PS - 3, 2/3	29,6819	357	3308	03:05:35	22,9	490,3	0,049	34,52	34,16

GPSPositiveBasisNp1, GPSPositiveBasis2N, GSSPositiveBasisNp1, GSSPositiveBasis2N, MADSPositiveBasisNp1, MADSPositiveBasis2N, GA search, Nelder-Mead search and Latin hypercube search were subsequently added as search methods to the regular PS and tested, and the results can be seen in Table 4.5. The relative differences to the GA reference method in terms of achieved objective function value and time to complete the optimization simulation needed to achieve said value are visualized in Figure 4.4 and Figure 4.5. As can be seen from the table and figures there are relatively small differences in value of objective function. However, even small improvements in the solution can be of significance when there are large sums of money involved. All the PS searches except GSSPositiveBasis2N arrived at a better solution than the reference GA method, and all searches except the two MADS searches took less time in doing so.

Table 4.5 – Comparison of various search methods added to regular pattern search

	Objective function value	Iter	F-count	Time [h]	LSP [bar]	LST [°C]	Pcond [bar]	PPTD [°C]	DPHRSG [mbar]
Regular pattern search	29,6819	357	3308	03:05:35	22,9	490,3	0,0485	34,52	34,16
GPSPositiveBasisNp1	29,6703	93	901	00:50:26	24,4	490,7	0,0487	35,00	34,94
GPSPositiveBasis2N	29,6819	357	3308	03:06:03	22,9	490,3	0,0485	34,52	34,16
GSSPositiveBasisNp1	29,6703	93	884	00:49:18	24,4	490,7	0,0487	35,00	34,94
GSSPositiveBasis2N	29,7300	71	718	00:40:00	24,4	490,8	0,0542	35,00	34,94
MADSPositiveBasisNp1	29,6819	357	5023	04:41:49	22,9	490,3	0,0485	34,52	34,16
MADSPositiveBasis2N	29,6708	257	3816	03:33:23	22,9	490,4	0,0485	33,56	34,94
GA search	29,6688	182	2779	02:35:36	24,3	487,3	0,0485	33,06	35,00
Nelder-Mead search	29,6819	357	3573	03:20:00	22,9	490,3	0,0485	34,52	34,16
Latin hypercube search	29,6701	211	1878	01:47:19	22,9	490,4	0,0486	35,00	34,82

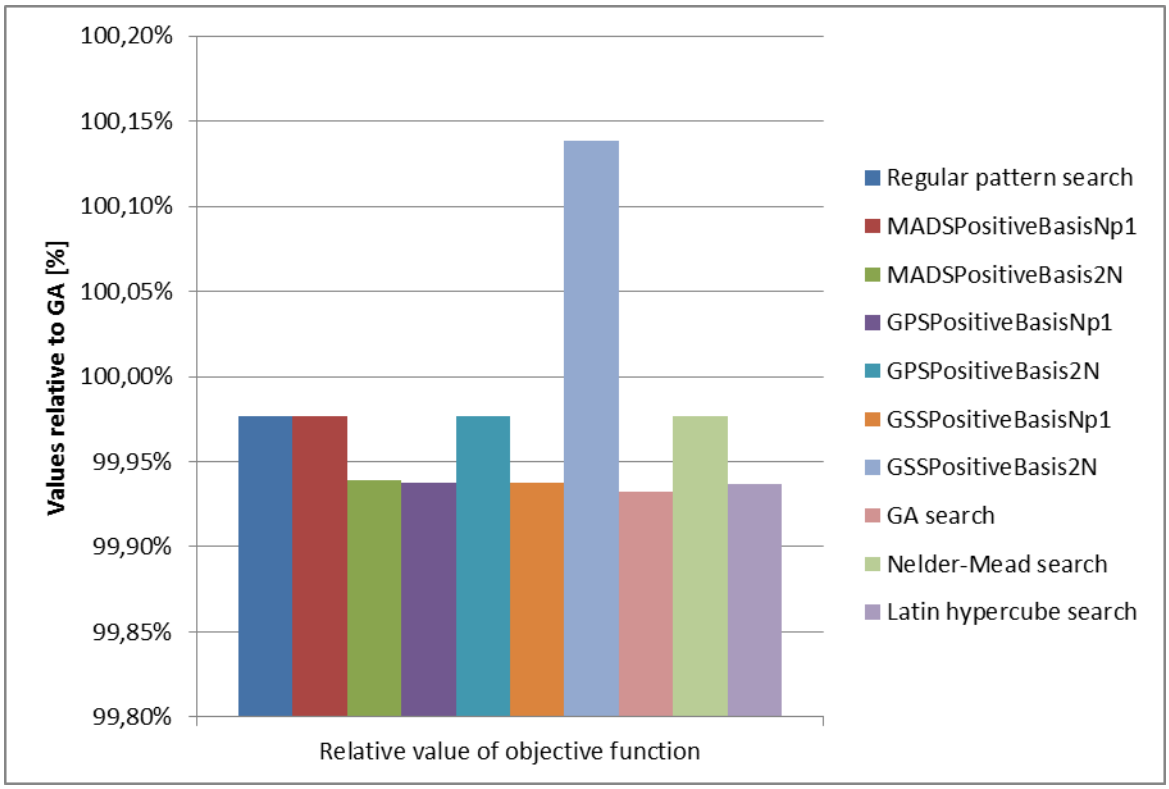

Figure 4.4 – Values of objective function relative to a GA reference case for PS with added search methods.

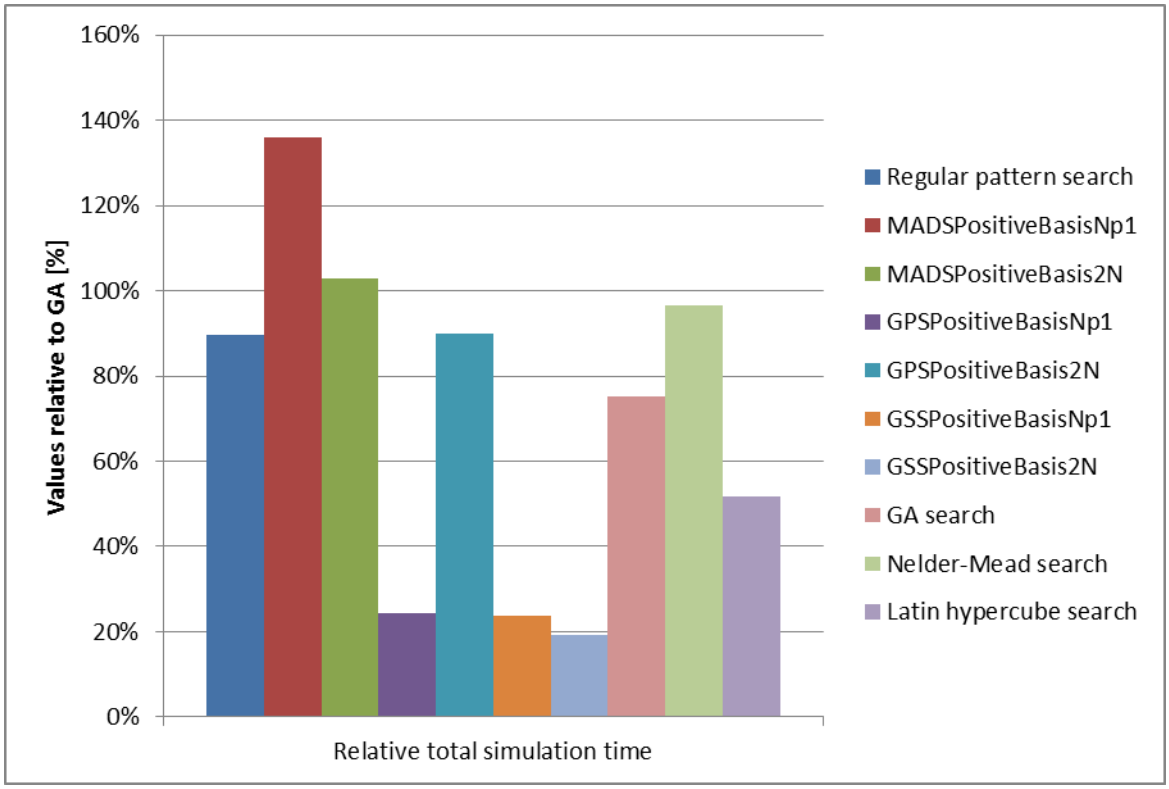

Figure 4.5 - Computation time for PS with added search methods relative to a GA reference case.

As one would expect based on theory, adding a GPSPositiveBasis2N search to a PS using a GPSPositiveBasis2N polling pattern makes absolutely no difference, one might as well just use the regular PS. Further, the GSSPositiveBasisNp1 search does indeed appear to perform identical to the GPSPositiveBasisNp1 search, as one would expect for an objective function without constraints. However, surprisingly the GSSPositiveBasis2N search was nothing like the GPSPositiveBasis2N search, though perhaps the GPSPositiveBasis2N search not really functioning when it is also used as the polling method had a part to play in that. In any case neither of the GSS searches appeared to be particularly interesting to analyse any further. Perhaps more surprisingly, MADSPositiveBasisNp1 appeared to be identical to the regular PS in terms of what solution it arrived at, and how many iterations it took getting there, though with significantly more objective function evaluations and consequently significantly longer total simulation time. For some reason it seemed like a very poor fit with a GPSPositiveBasis2N poll method. Ignoring the similar searches; in terms of arriving at the best solution MADSPositiveBasis2N, GPSPositiveBasisNp1, GA search and Latin hypercube search appeared to have performed the best, while GPSPositiveBasisNp1, GSSPositiveBasis2N and Latin hypercube search took the least time in doing so. With GSSPositiveBasis2N being of little interest due to its poor solution, the most interesting search methods remaining appeared to be the MADSPositiveBasis2N search , the GPSPositiveBasisNp1 search, the GA search and the Latin hypercube search.

Further, as a MADS search in theory should be a good fit with a GPS poll it is somewhat surprising that a GPSPositiveBasisNp1 search located a better solution than a MADSPositiveBasis2N search. Generally one would have expected the MADSPositiveBasis2N search to arrive at the best solution, though perhaps the settings for this initial screening were particularly suitable for the GPSPositiveBasisNp1 search, or less favourable to the MADSPositiveBasis2N search. It was believed to warrant a second slightly more thorough comparison of the two methods with different mesh expansion/contraction and without a complete poll.

4.3.2 Second Screening

Considering achieving the best possible solution is the most important aspect of the optimization process, the options structure in the second screening between GPSPositiveBasisNp1 search and MADSPositiveBasis2N search was modified in an attempt to locate an improved solution. The expansion and contraction factor were adjusted to have the methods evaluate more points, and as such cover more space of the interesting search region. This meant increasing the expansion factor for the mesh, and as such increasing the size of the mesh after each successful poll and cover more space, as well as decreasing how much the mesh is reduced after an unsuccessful poll, thereby checking the area surrounding the current point more thoroughly. Consequently, the contraction factor was set to 0.8 for all the attempted searches at this screening, while the expansion factor was tried as both 3 and 4. Due to the uncertainties regarding the impact of the complete poll setting, the searches were also carried out both with and without a complete poll. However,

unfortunately the MADSPositiveBasis2N searches with an expansion factor of 4 crashed repeatedly with the complete poll turned off, and with the MADSPositiveBasis2N search crashing it seemed rather uninteresting to consider the GPSPositiveBasisNp1 with the complete poll turned off, particularly as there appeared to be little difference between having it turned on or off for the GPSPositiveBasisNp1 search. As a consequence the searches with an expansion factor of 4 only include the complete poll setting set to on.

The result of this second screening can be seen in Table 4.6 and Figure 4.6 and Figure 4.7. The increased contraction factor did as expected result in a better solution for the MADSPositiveBasis2N search. However, for the GPSPositiveBasisNp1 search the solution was actually noticeably worse, lending credibility to the theory that it perhaps was a bit fortunate with the settings in the initial screening. The MADSPositiveBasis2N search experienced a further improvement in the solution with an increased expansion factor, now with a better solution than any of the search methods achieved in the initial screening. Increasing the expansion factor also significantly improved the GPSPositiveBasisNp1 search for the same contraction factor, but was still worse than in the initial screening, only reinforcing the impression that it was somewhat lucky in the first screening. In terms of computational time, it increased markedly both for increased expansion factor and for an increased contraction factor, as one would expect. As for the complete poll, the GPSPositiveBasisNp1 search experienced little if any difference whether it was turned on or off, while the MADSPositiveBasis2N search was seemingly significantly quicker without a complete poll, as well as ending up with a marginally better solution.

Table 4.6 – Second screening, comparison between GPSPositiveBasisNp1and MADSPositiveBasis2N

	Expansion factor	Contraction factor	Complete poll	Objective function value	Iter	F-count	Time [h]
GPSPositiveBasisNp1	3	0,8	off	29,6829	179	2271	02:07:21
GPSPositiveBasisNp1	3	0,8	on	29,6829	179	2292	02:09:05
MADSPositiveBasis2N	3	0,8	off	29,6699	309	3556	03:18:32
MADSPositiveBasis2N	3	0,8	on	29,6699	363	4555	04:14:13
GPSPositiveBasisNp1	4	0,8	on	29,6735	400	5618	05:12:51
MADSPositiveBasis2N	4	0,8	on	29,6685	569	8702	08:07:04

Being unable to complete a PS with MADSPositiveBasis2N search, with an expansion factor of 4 and the complete poll turned off, without the computer experiencing a blue screen error was a bit of a nuisance. Nonetheless, this second screening suggested it might be difficult for the GPSPositiveBasisNp1 search to improve upon the results it achieved in the initial screening. The MADSPositiveBasis2N search on the other hand improved both its solution and total simulation time with the contraction factor increased to 0.8 and the complete poll turned off, and showed further improvement in the solution when the expansion factor was increased. With that in mind, and the fact that the

MADSPositiveBasis2N search according to the theory should be a better fit for the default polling method, it seemed logical to proceed with the MADSPositiveBasis2N search at the expense of the GPSPositiveBasisNp1 search.

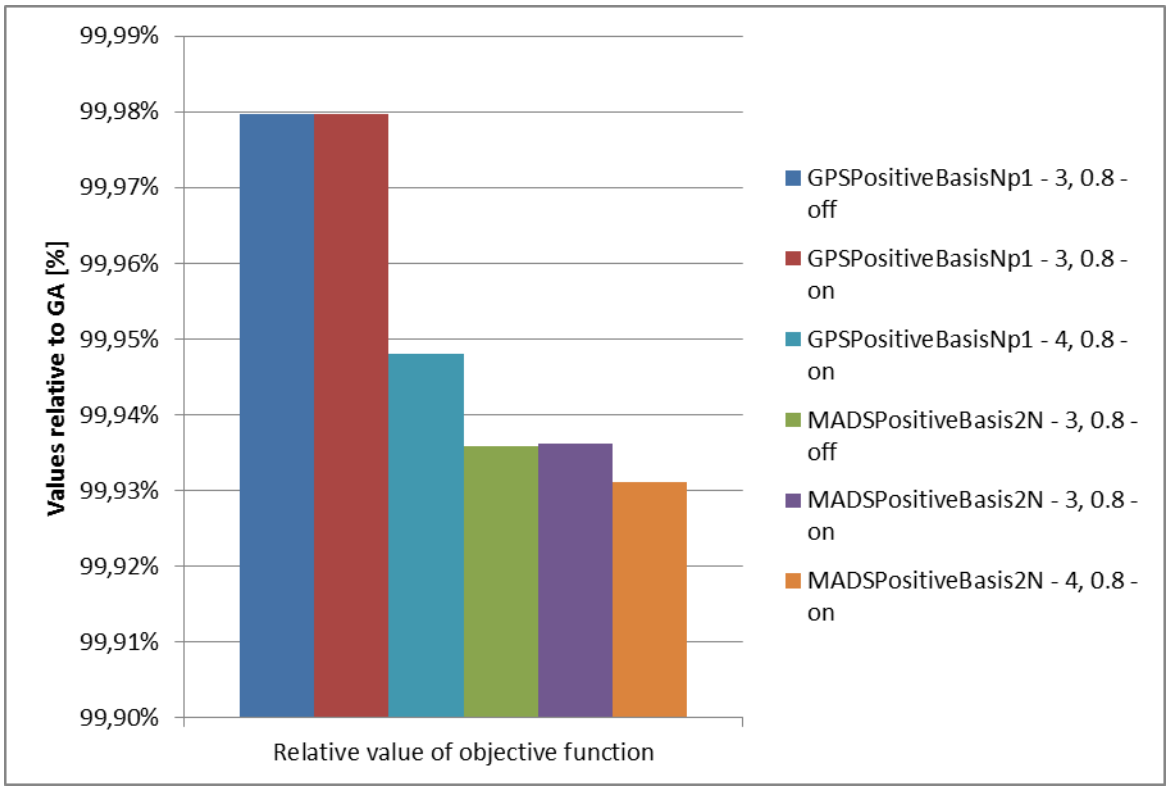

Figure 4.6 - Values of objective function in second screening, relative to GA reference case.

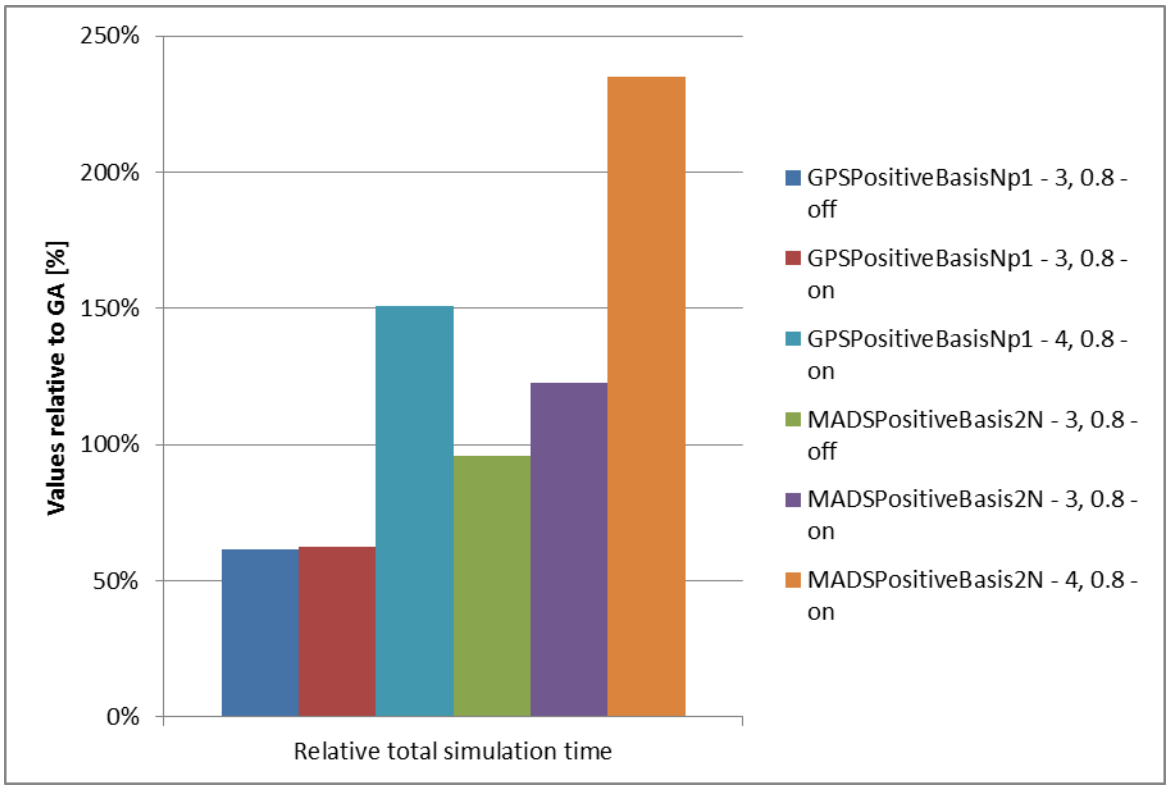

Figure 4.7 - Computation time second screening relative to GA reference case.

As such, after two separate screening processes the interesting search algorithms for the PS was narrowed down to the MADSPositiveBasis2N and the fundamentally different GA and Latin hypercube search. With the regular GA remaining as a reference method, and the MADSPositiveBasis2N search being fundamentally different from the two other potential PS search methods, that left deciding between which of the GA search and the Latin hypercube search to move on with.

In the initial screening, the GA search arrived at a slightly better solution than the Latin hypercube search, though took a fair bit of extra time in doing so. However, taking everything into account they did seem fairly interchangeable. Considering both methods are identical after the initial search for a better objective function value, there was little reason to suspect them to deviate much at any point if any of the other PS options were to be adjusted, and as such it was assumed to be a poor use of resources to perform an additional screening between the two. However, an interesting aspect with the GA search was that it to some extent could be considered as a hybrid solution between the two other searches retained for further analysis, as it essentially uses the GA to search for a good starting position to start a regular PS. With that in mind, and the fact that the GA search ultimately did obtain a better solution in the initial screening, it was decided to retain the GA search. Hence, the final three optimization methods to be explored further were the GA as a reference method, the PS with a MADSPositiveBasis2N search and the PS with GA search.

4.4 Modifications of Selected Optimization Methods

As previously alluded to there were some issues with the computer crashing during some of the simulations. The crashes seemed more rampant the longer the simulations went on, making finding the best possible solution particularly challenging. Setting parameters like populations size, mesh expansion or mesh contraction too large seemingly always guaranteed the simulation to crash. However, on rare occasions a simulation that had previously crashed several times would all of a sudden work flawlessly. Not knowing when a simulation might crash made completing systematic searches a challenge, particularly for the very long simulations that with the regular crashes quickly became a protracted affair. Eventually had to balance between trying to achieve the very optimal solution and working within the confines of what would usually allow the simulations to work flawlessly. The frequent crashes for long lasting simulations also meant that trying to modify the stopping criteria in the form of reducing the mesh tolerance, and hence prolonging the searches, was never really entertained as an option. Neither was applying more stringent stopping criteria causing the searches to stop sooner. Having the default mesh tolerance as the most restrictive stopping criteria was believed to offer the best compromise of accuracy of the solution and time to reach it, as well as giving all the search methods equal chance at locating the best possible solution by having similar size of the mesh in their last polls of a search.

4.4.1 Pattern Searches

For the pattern searches the previous screenings had shown that modifications to the mesh expansion and contraction as well as whether a complete poll was used could have a significant impact on the performance of the algorithms. As explained previously, increasing the mesh expansion should allow the algorithm to cover more of the solution space, and hence being able to find a better solution than with a smaller expansion factor, though taking more time in doing so. Increasing the contraction factor should have much of the same effect, though rather than covering more of the solution space it would mean searching the region surrounding the current point more thoroughly. Additionally it was thought that perhaps more stringent boundaries on the design parameters could reduce the simulation time. After all the lower bounds used in the initial screenings did contain some regions very unlikely to contain a good solution. As discussed previously, it was very unlikely that any decent solution would have a LST below 455 °C, while a reduction in the PPTD or HRSG draft loss from the project work values seemed equally unlikely. Consequently, when the boundaries were adjusted the lower boundaries for those three parameters were narrowed down to respectively 455 °C, 20 °C and 25 mbar.

Table 4.7 and Figure 4.8 shows the most interesting successful simulations of the PS using a MADSPositiveBasis2N search algorithm, and the adjustments made to the options structure of said searches. As can be seen from the second row of the table and the second column of the figure, increasing the contraction factor does indeed result in the algorithm finding a better solution, as well as increasing the simulation time. As discovered in the previous

screening, turning the complete poll off actually has a positive effect on both the achieved solution and the simulation time. However, somewhat surprisingly, as seen from the 4th row of the table and 4th column of the figure, narrowing the LB actually had an adverse effect on the solution found, though the algorithm did arrive at it more quickly. Then countless simulations with an expansion factor of 4 with complete poll set to off were attempted to no avail; as in the second screening they all crashed, usually after about six hours. Then in an act of despair an expansion factor of 5 was attempted, which inexplicably worked on the second attempt. Perhaps only a stroke of good luck, but in any case the results it yielded were as one would expect. The simulation time obviously was significantly longer than any of the other simulations performed so far, almost eclipsing 13 hours. Fortunately the improvement of the solution seemingly matched the extra computation time, and it was a slight improvement on the result achieved with an expansion factor of 4 and the complete poll set to on. A further increase in contraction factor to 0.9 was also attempted, with an expansion factor varying from 3 to 5, though with no successful simulations.

Table 4.7 – PS with MADSPositiveBasis2N search, changes to options structure

	Expansion factor	Contraction factor	LB changed	Complete poll	Objective function value	Iter	F-count	Time [h]
Initial screening	3	0,67	no	on	29,6707699	257	3816	03:33:23
3, 0.8, no, on	3	0,8	no	on	29,6699491	363	4555	04:14:13
3, 0.8, no, off	3	0,8	no	off	29,6698521	309	3556	03:18:32
3, 0.8, yes, off	3	0,8	yes	off	29,6720417	267	3052	02:50:18
4, 0.8, no, on	4	0,8	no	on	29,668471	569	8702	08:07:04
5, 0.8, no, off	5	0,8	no	off	29,6681832	876	13291	12:51:52

Looking a bit more in depth on the PS with a MADSPositiveBasis2N search algorithm with an expansion factor of 5, contraction factor of 0.8, no adjustments to LB and without a complete poll, Figure 4.9 shows how the value of the objective function evolved over the iterations. It quite rapidly, after 129 iterations and 1168 function evaluations, achieved a quite decent objective function value of 29.6728, which was followed by a series of incremental improvements until the mesh size was smaller than the mesh tolerance, culminating in an objective function value of 29.6682.

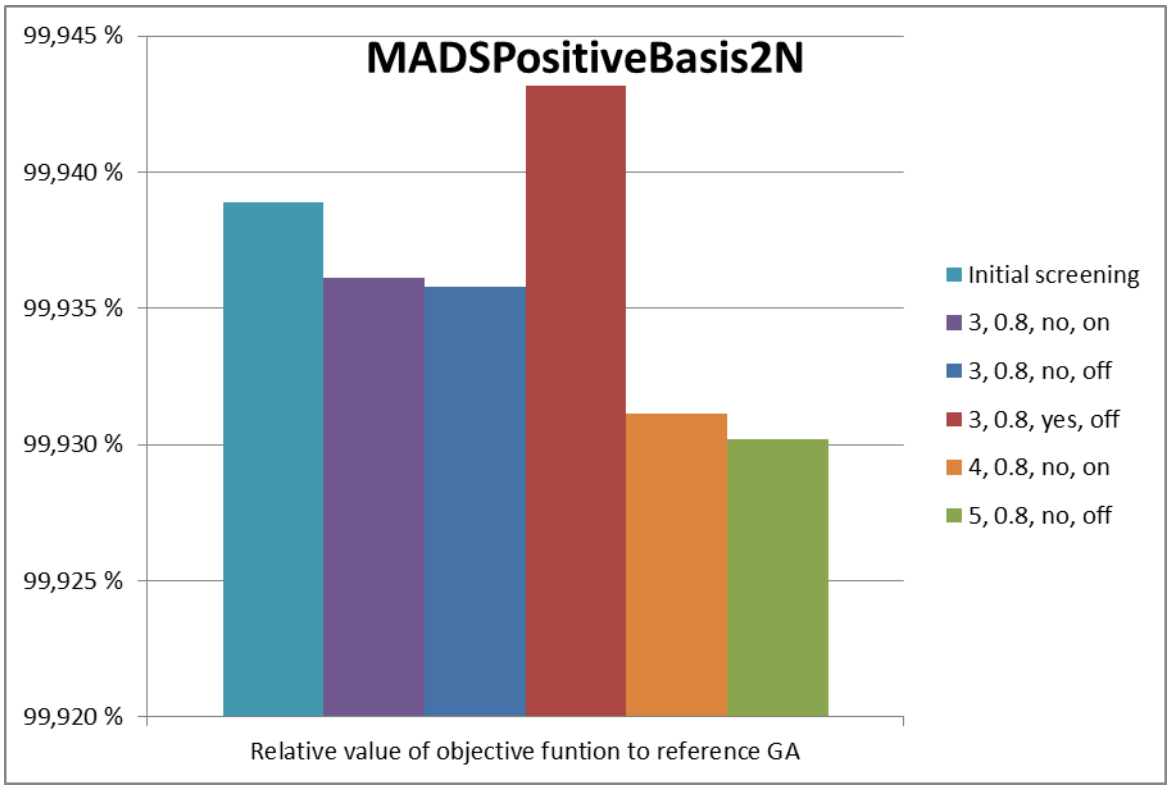

Figure 4.8 – PS with MADSPositiveBasis2N search, objective function values after changes to options structure.

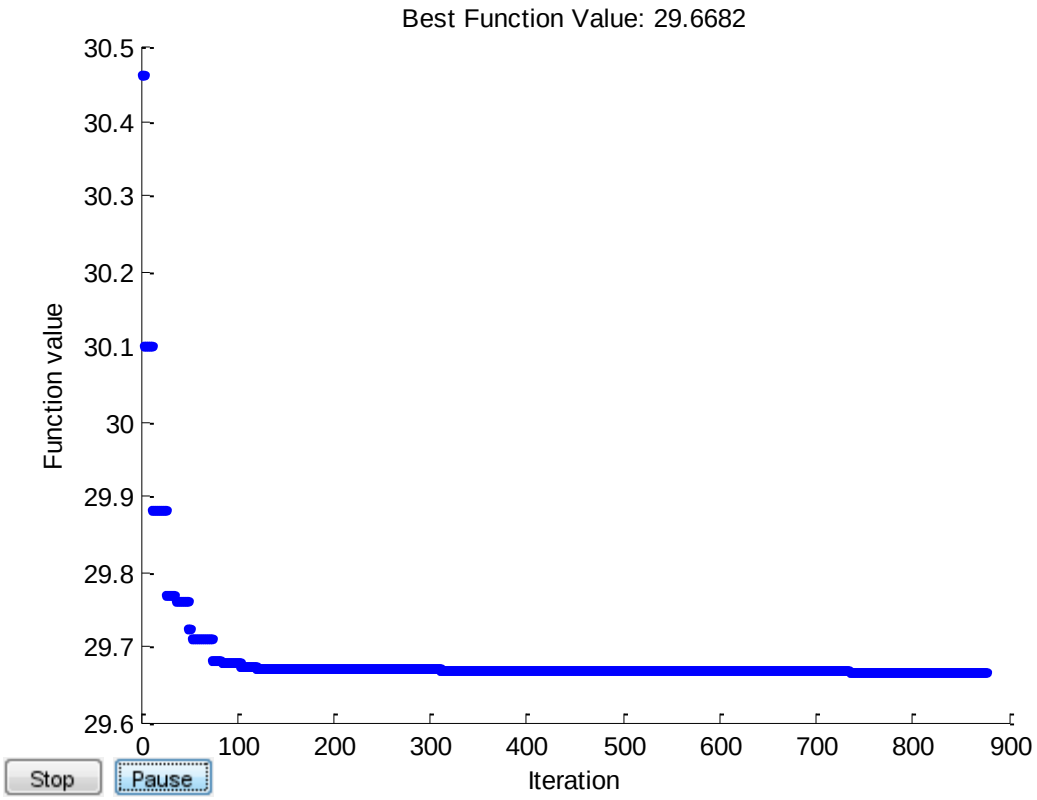

Figure 4.9 - PS with a MADSPositiveBasis2N search algorithm with an expansion factor of 5, contraction factor of 0.8, no adjustments made to LB and without a complete poll.

With the changes in LB a bit surprisingly having very little impact on the results, a couple of more tests with both stricter and more relaxed upper and lower boundaries were carried out, using the settings from the initial screening with a PS with a MADSPositiveBasis2N search algorithm. However, the upper limit of the PPTD and DPHRSG was maintained at the same level as that is right around where the best solution without frequent crashes seemingly occur. The upper boundary on the LST was also retained, as when it was attempted increased to about 550 °C the optimization returned a solution where the LST was higher than that of the gas turbine exhaust temperature, which is clearly an infeasible solution. As seen from Table 4.8, a tightening of the boundaries actually increased the simulation time as well as somewhat worsening the final solution. The more relaxed boundaries reduced the simulation time, but ended up with a similar solution to that of the search with tighter boundaries. It seems as if the upper and lower boundaries have fairly little impact on the performance of a PS with a MADSPositiveBasis2N search algorithm, if anything the initially chosen boundaries seem to produce the best results.

Table 4.8 - Pattern search with MADSPositiveBasis2N search algorithm with varying upper and lower boundaries

	LB [LSP, LST, Pcond, PPTD, DPHRSG]	UB [LSP, LST, Pcond, PPTD, DPHRSG]	Objective function value	Iter	F-count	Time [h]
Initial screening	[15 400 0.0234 8 15]	[35 510 0.14 35 35]	29,6707699	257	3816	03:33:23
Tighter boundaries	[20 455 0.04 24 29]	[30 510 0.10 35 35]	29,6740477	342	5455	05:08:59
More relaxed boundaries	[10 355 0.0234 5 5]	[55 510 0.20 35 35]	29,6739821	216	2729	02:26:40

With the change in upper and lower boundaries having little effect on the outcome of the search, it was assumed that part of the explanation for that could be down to how the PS works. As the PS focuses on exploring the region immediately surrounding the current point, and the current point is typically relatively close to a good solution for the most of simulation time, it will rarely investigate locations far from a good solution. Even if the initial point is located in a region with a poor solution, and far from the optimal solution, the current point will move quite rapidly towards a better solution, so the time spent investigating the region of a poor solution will be short. In an attempt to confirm this hypothesis, a small test using the settings from the initial screening with a PS with a MADSPositiveBasis2N search algorithm with varying initial points was carried out. As seen from Table 4.9, there was very little difference resulting from different initial points. In fact, the search using the values from the project work actually achieved both the worst solution and took the longest time in doing so, despite having by far the best objective function value at the initial point. The simulation starting the furthest from the best solution, listed as the 3rd search in the table, actually took the shortest time in finding its solution. The differences

are essentially negligible though. Overall it seems as if the selection of the initial point is fairly insignificant for a pattern search with a MADSPositiveBasis2N search algorithm.

Table 4.9 – Pattern search with MADSPositiveBasis2N search algorithm with varying initial point

Initial point [LSP, LST, Pcond, PPTD, DPHRSG]	Objective function value	Iter	F-count	Time [h]
x0 = [25 470 0.08 25 30]	29,6707699	257	3816	03:33:23
x0 = [15 400 0.03 8 15]	29,6695171	266	3479	03:16:04
x0 = [15 400 0.13 8 15]	29,6697019	238	2890	02:42:58

To test is this was true for other pattern searches as well, and not just an anomaly when using a MADSPositiveBasis2N search algorithm, a similar test was performed with a regular patter search. That test yielded very similar results, suggesting that the selection of the initial point might actually be of very little importance for pattern searches.

For the PS with a GA search it was attempted to employ roughly the same adjustments to the options structure as for the PS with the MADSPositiveBasis2N search. The GA search had much of the same crash issues, particularly with a contraction factor of 0.9, but unlike the MADSPositiveBasis2N search one of the simulations with an expansion factor of 4 was actually successful. However, any changes made to the options structure had seemingly very little bearing on the final result, as seen from both Table 4.10 and Figure 4.10. Actually none of the adjustments resulted in an improvement upon the solution from the initial screening, though the simulation with an adjustment made to the LB was the closest. This is perhaps indicating that the boundaries set are of some importance for the GA search, which seems reasonable considering the default GA searches the entire solution space.

Table 4.10 - PS with GA search, changes to options structure

	Expansion factor	Contraction factor	LB changed	Complete poll	Objective function value	Iter	F-count	Time [h]
Initial screening	3	0,67	no	on	29,6687982	182	2779	02:35:36
3, 0.8, no, on	3	0,8	no	on	29,6692836	354	4354	04:05:44
4, 0.8, no, on	4	0,8	no	on	29,6694707	446	5119	04:46:16
5, 0.8, no, on	5	0,8	no	on	29,6693091	392	3824	03:34:21
5, 0.8, no, off	5	0,8	no	off	29,6694709	466	5026	04:41:26
3, 0.8, yes, off	3	0,8	yes	off	29,669022	289	2879	02:40:51

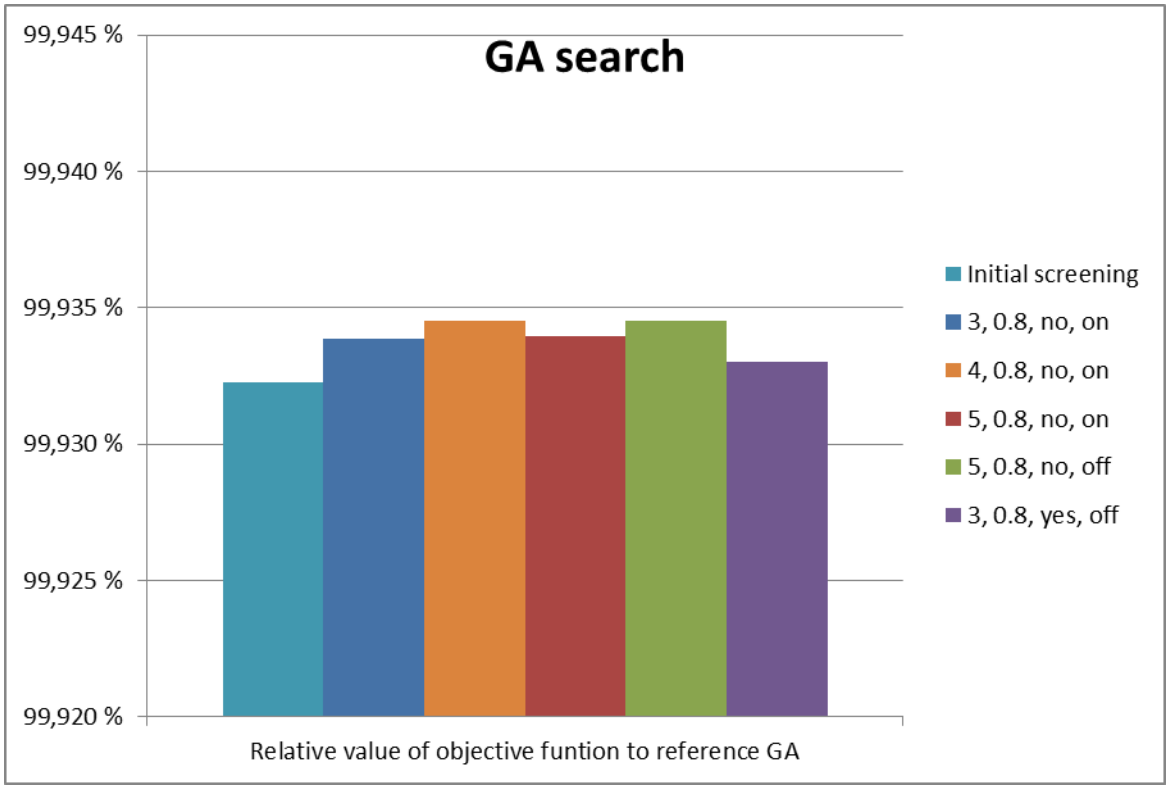

Figure 4.10 – PS with GA search, objective function values after changes to options structure.

Looking a bit closer on the details of the simulations, it appears as if the initial screening was perhaps a bit lucky with the initial GA search. As discussed previously, using a GA search with a PS is more or less equivalent with choosing a good starting position for the regular PS, and the initial screening managed to locate a point with an objective function value of 29.7437, much better than the other GA searches that were in the range of 29.85. There were no changes made to the GA part of the search, so the difference must be attached to stochastic nature of the GA. Looking at Figure 4.11 and comparing it to Figure 4.9 it is evident that the GA search managed to locate a point after the first iteration that it took the MADSPositiveBasis2N search about 50 iterations to achieve. Further, as the GA search did not need to contract its mesh size in order to locate said point, it had an advantage in that it with a relatively larger mesh size at that point quite quickly could locate further improvements to the objective function. However, it took the GA search a not insignificant 1151 function evaluations to complete the first iteration. A number of function evaluations that the PS with the MADSPositiveBasis2N search only reached after 128 iterations. However, with the relatively large mesh size of the PS after the GA search has run its course, the algorithm can locate better points quite rapidly, possibly making up for any disadvantages in terms of number of function evaluations.

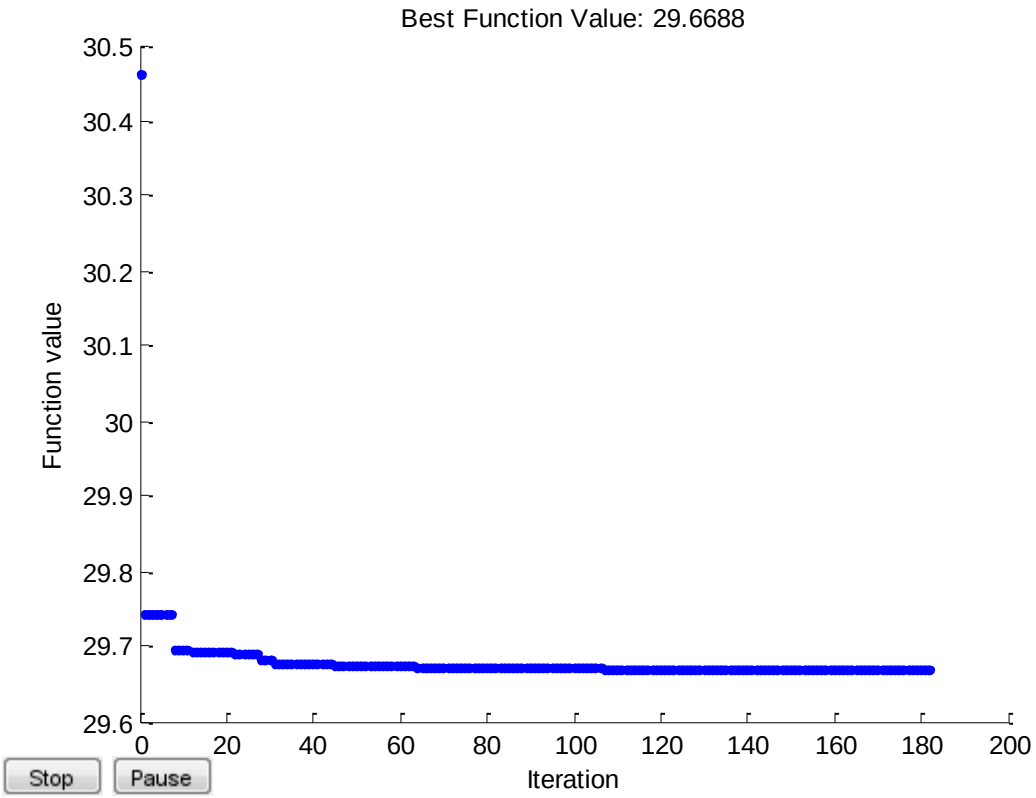

Figure 4.11 – PS with GA search from initial screening, expansion factor of 3, contraction factor of 0.67, no adjustments to LB but with a complete poll.

To test if the changes in boundaries were of some significance with a GA search, a similar test to what was performed for the PS with MADSPositiveBasis2N search algorithm was carried out. As seen from Table 4.11 does seem to have some impact. The search with more relaxed boundaries was noticeably worse. However, the tighter boundaries did not result in a better solution than the initial screening. As such, it appears as if the best approach is a boundary that is neither too relaxed nor too tight, and that the initial boundary selected was quite good. However, as previously speculated, it is possible that the initial screening was a bit lucky, and that for several simulations over time a tighter boundary might lead to better results on average.

Table 4.11 - Pattern search with GA search algorithm with varying upper and lower boundaries

	LB [LSP, LST, Pcond, PPTD, DPHRSG]	**UB [LSP, LST, Pcond, PPTD, DPHRSG]**	**Objective function value**	**Iter**	**F-count**	**Time [h]**
Initial screening	[15 400 0.0234 8 15]	[35 510 0.14 35 35]	29,6687982	182	2779	02:35:36
Tighter boundaries	[20 455 0.04 24 29]	[30 510 0.10 35 35]	29,6752004	197	2571	02:24:44
More relaxed boundaries	[10 355 0.0234 5 5]	[55 510 0.20 35 35]	29,7002988	133	2237	02:05:23

Considering the purpose of using a GA search with a PS is to improve the initial point for the PS, it was assumed to be unnecessary to test different initial points for the search.

4.4.2 Genetic Algorithm

The GA is quite a bit different from the two other searches, and though it was initially thought of as a reference model to validate that the PS algorithms located sensible solutions, it was thought to be of interest to also try to improve its performance. However, even with the difference from the PS algorithms, GA also had its fair share of problematic crashes, with its Achilles' heel mainly being the size of the population. Any increase in the population size from the 100 used in the initial screening seemed futile. Still, there were other adjustments that could be made.

By default the GA uses an elite count of only 2, as for a smooth objective function a lower elite count would result in the optimum being achieved quicker. However, the objective function being optimized in this thesis appeared to be laden with local minima. As such it was thought that an increase in the elite count could allow the GA to explore more local minima at each generation, possibly resulting in a better solution. Another aspect was that as the GA randomly searches the entire solution space, it would probably stand to benefit greatly from a narrowed solution space to search. As discussed in the theory, the initial population range permits creating a range from which the design parameters used to create the initial population is selected from, while still allowing the algorithm to potentially find solutions outside of the specified region. As such it adds the benefits of a narrower solution space, but without permanently removing the suspected uninteresting parts of it. It also renders having an initial population somewhat meaningless. Lastly there is the matter of the stopping criteria of the algorithm, which with the default settings typically is the function tolerance. To try and prolong some of the searches, the function tolerance was attempted reduced, and when it was set to zero it prompted the need for an alternative stopping criteria. The number of generations was thought to be a convenient stopping criterion, and the default value of 100 seemed suitable enough. Adding a narrower LB as was done for the pattern searches was also attempted, though it seemed a bit superfluous with an initial range already being used.

As for deciding on what values to use for the initial population range, the design variables from the project work was used as a basis, and then a plus and minus range was added to the project work value. For the LSP 5 bar was added and subtracted from the 25 bars in the project work, for the LST there was added and subtracted 30 °C to the initial 470 °C while for the condenser pressure 0.04 bar was added and subtracted. PPTD and DPHRSG were a bit different as they were thought to be very unlikely to go down, but the lower range limit was still set 5 °C and mbar below the project work respectively, while the maximum was set to what was believed to be the maximum possible without running into frequent computation errors. As such design variable range for the initial population creates was 20 - 30 bar for the

LSP, 440 – 510 °C for the LST, 0.04 – 0.12 bar for the condenser pressure, 20 - 33.3 °C for the PPTD and 25 – 35 mbar for the DPHRSG.

Table 4.12 contains various GA simulations where adjustments have been made to the default options structure, and the differences in obtained objective function value are visualized in Figure 4.12. It is worth noting that the initial screening is not included in graph as all columns are relative to that of the initial screening. It appears as if increasing the elite count from 2 to 5 in order to explore more minima simultaneously has a positive impact on the solution attained, though a further increase to 15 seemed a bit more risky. One simulation with 15 as elite count ended up with the best solution of any of the GA optimizations, while another simulation with exactly the same settings ended up with a worse solution than any of the GA optimizations apart from the initial screening. As such, an elite count of 5 might seem like the safest approach to obtain a good solution for this problem. Decreasing the function tolerance does as one would expect increase the simulation time, and allows the algorithm time to locate a better solution, as seen quite evidently when comparing column 2 with column 3 and column 5 of Figure 4.12. Further, as seen from the comparison of column 1 and column 2 and column 4 and 5 of the same Figure 4.12, adding a range for the initial population does indeed help in regards to ending up with a better solution. The impact of adding an initial range also seems to be of more importance for shorter simulations, which seems logical as shorter searches are probably more dependent on having a quite good initial population to work with as it has less time to recover from any potential poor population composition. As expected, adding a narrowed LB made very little difference, it actually resulted in a worse solution. Using an initial range for the initial population seems much more beneficial.

Table 4.12 - GA, various changes to options structure

	Elite count	Initial range	Initial population	Function Tolerance	Objective function value	Iter	F-count	Time [h]
Initial screening	2	no	yes	1e-7	29,6889125	36	3700	03:27:08
5, no, no, 1e-7	5	no	no	1e-7	29,6778215	28	2900	02:42:50
5, yes, no, 1e-7	5	yes	no	1e-7	29,6751831	27	2800	02:37:36
5, yes, no, 1e-9	5	yes	no	1e-9	29,6719687	88	8900	08:17:21
5, no, no, 0	5	no	no	0	29,6701337	100	10100	09:27:58
5, yes, no, 0	5	yes	no	0	29,6693852	100	10100	09:41:42
15, yes, no, 0	15	yes	no	0	29,6686772	100	10100	09:25:16
LB, 5, yes, no, 0	5	yes	no	0	29,6703843	100	10100	09:25:55

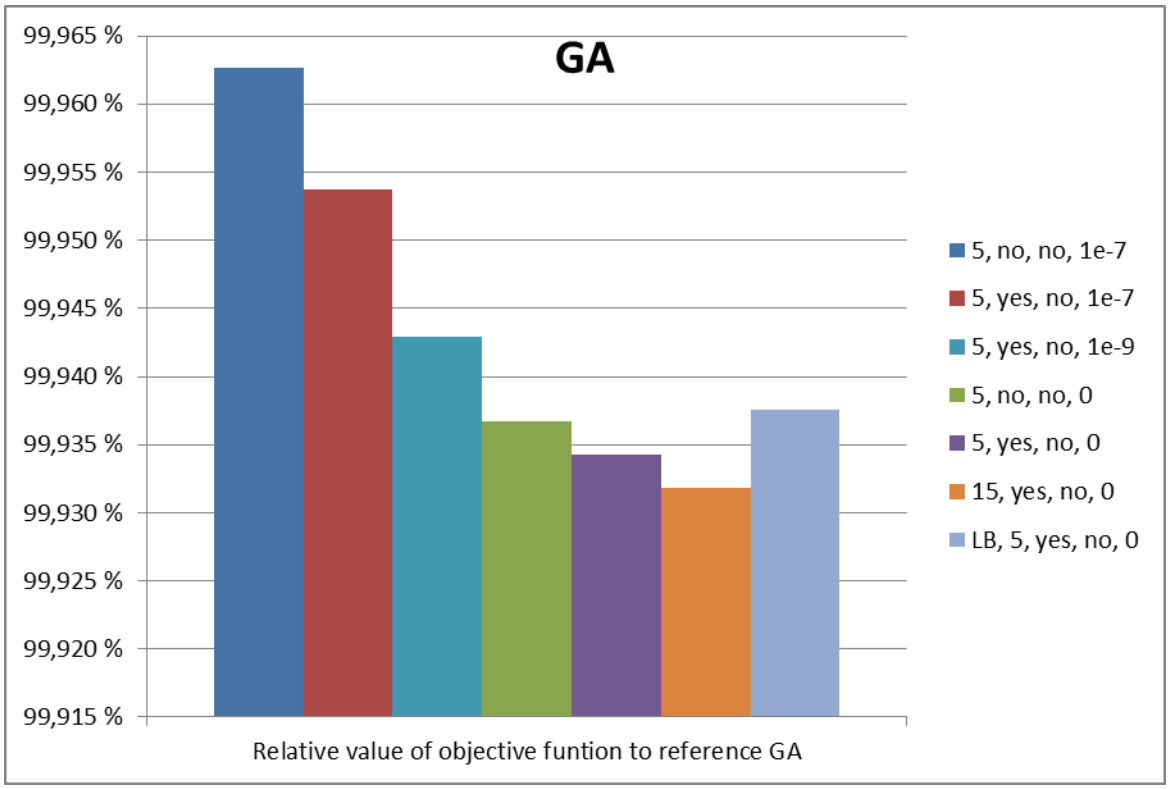

Figure 4.12 - GA, objective function values with changes to options structure.

As discussed in the theory the GA does have an element of randomness to it. This was illustrated by the first generation usually finding an individual with a fitness value around 30.0, though for some simulations the best individual in the first generation had a fitness value closer to 31.0, or in other words a worse solution than the project work. However, given enough simulation time it ended up with more or less the same final solution. Though for short simulations, like the ones with a function tolerance around the default value of 1e-7, algorithms with identical settings could end up with relatively vastly different solutions from different simulations.

Figure 4.13 shows the development of the best fitness vale and the mean value for the iterations of the GA arriving at the best GA solution. However, the mean values are slightly misleading. As the objective value is set to 50 when the calculations in GT PRO contains warning messages, populations with a lot of erroneous calculations can have a high mean fitness value even when the rest of the population develops towards a better fitness value. The graph does paint a picture of how many of the calculations actually were problematic though, especially as none of the individuals with a warning message would be particularly likely to survive to the next generation. Considering that with the initial range selected most individuals in a typical population would have a fitness value around 30, there would need to be around 25 individuals with a fitness value of 50 for the mean in a population of 100 to be as high as 35. That is a staggering 25 % of the simulations, and possibly a part of the reason why so many of the long lasting simulations seemingly crashed. In regards to how the individual with the best fitness value evolved, it quite quickly reached a fitness value of

around 29.75, before almost 40 iterations with only marginal improvements before a significantly better individual was discovered, followed by a series of very small incremental gains before stalling completely over the last 10 generations.

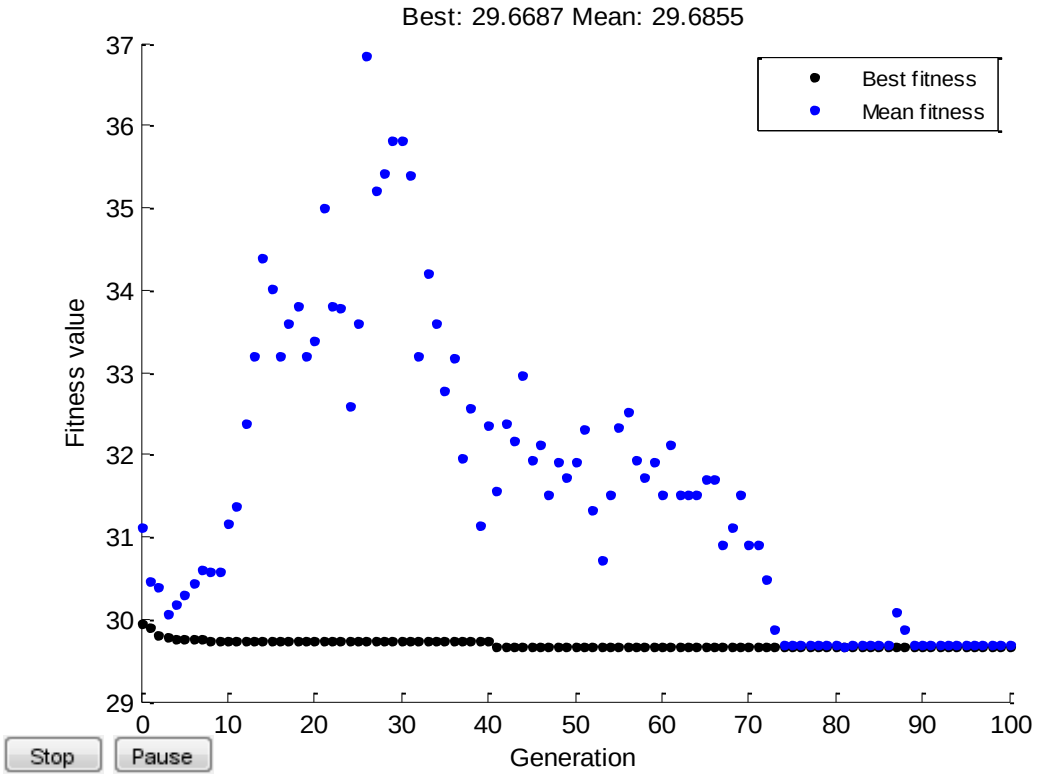

Figure 4.13 – GA with an elite count of 15, selected range for initial population, no selected individual in the initial populations and 0 function tolerance. Be aware that the mean values of the population are slightly misleading.

On a whole, the differences between the various optimization methods seemed to be pretty minor. They all arrived at roughly the same solution, and though some were quicker in arriving at its solution, the total simulation time was relatively low to begin with, in essence making the difference in simulation time relatively insignificant. However, if there had been no issues with simulations crashing, allowing for much longer simulations in general, then perhaps the simulation time would have been of more interest. Still, in terms of obtaining the very best possible solution, the pattern search with a MADSPositiveBasis2N algorithm search does look like a very attractive option. If a fairly good solution in a short time is desired, using a GPS or GSS search algorithm with the pattern search might be a decent choice. The pattern searches running a search algorithm in advance of the pattern search, like the GA search, seems to be able to achieve quite good solution in reasonable time, though it seems to struggle with finding a very good solution. Further, due to the random element of the GA algorithm, how fortunate the initial search is seems to have a greater impact on the final result than any changes to the options structure. The genetic algorithm is able to find a quite good solution with the proper adjustments to match the problem at

hand, though it is a bit random and has difficulties obtaining a very good solution, and it is not particularly quick either.

4.5 Best Optimized Solution

PS with MADSPositiveBasis2N search was able to locate the best solution, though the best solutions for all 3 algorithms were relatively close, and considerably better than the median of the initial screening as seen from Figure 4.14.

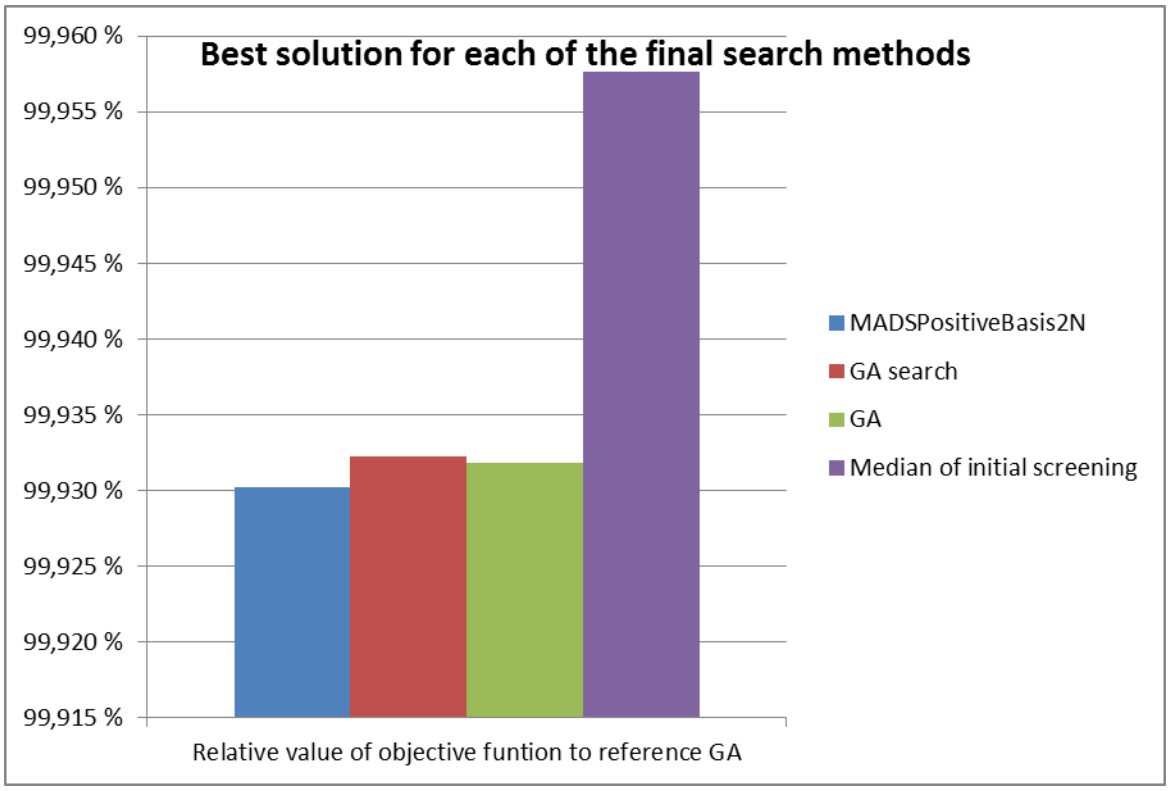

Figure 4.14 – Best solution of each of the final optimization methods and median of the initial screening, relative to the reference GA.

To check for further improvements another PS with MADSPositiveBasis2N search was performed, using the final values from the PS with a MADSPositiveBasis2N search algorithm with an expansion factor of 5 and contraction factor of 0.8 as the starting point for the search. After a search lasting almost 3 hours it arrived at the solution in Table 4.13, which is only a very marginal improvement on the previously best solution. As such it appears to be a solution that should be very close to the true global optimum.

Table 4.13 – PS with MADSPositiveBasis2N search from previously best solution

	Objective function value	LSP [bar]	LST [°C]	Pcond [bar]	PPTD [°C]	DPHRSG [mbar]
MADSPositiveBasis2N, 0.9, 3 from previously best solution	29,668165	24,347673	488,68725	0,0485562	33,251477	34,996414

It is also worth noting that for about 50 different simulations with an objective function value below 29.70, the difference in design variables were very minor, suggesting that the best solutions are all located in the same region. Table 4.14 shows the median, the lowest and the highest value of each of the design variables in those aforementioned 50 simulations, while Figure 4.15 illustrates the range of the design variables relative to the median values of those simulations. The only two parameters there appears to be some fluctuation for are the LSP and the PPTD. For the LSP it is worth noting that all but 3 of the simulations actually have a value below 24.7 bar, which is less than 2 % above the median, and hence skewing the positive range somewhat. As for the PPTD fluctuations, a lot can probably be attributed to issues with the GT PRO computations when the mass flux gets too large. When the DPHRSG is very close to the upper boundary set of 35 mbar, it appears as if the maximum PPTD attainable without relatively consistently having erroneous computations is around 33.3 °C, though there are occasionally some simulations that are successful even when both the PPDT and the DPHRSG variables attained are close to the upper limit set. Anyway, even with those fluctuations for the LSP and the PPTD, all the relatively good solutions are very close to each other, further strengthening the claim that the best solution obtain is likely to be very close to the true global optimum.

Table 4.14 - Range of design variables from simulations with an objective function value below 29.70

	LSP [bar]	LST [°C]	Pcond [bar]	PPTD [°C]	DPHRSG [mbar]
Median	24,28	490,35	0,04856	34,443	34,982
Lowest value	22,89	484,66	0,04738	32,947	34,159
Highest value	26,29	495,14	0,04881	35,000	35,000

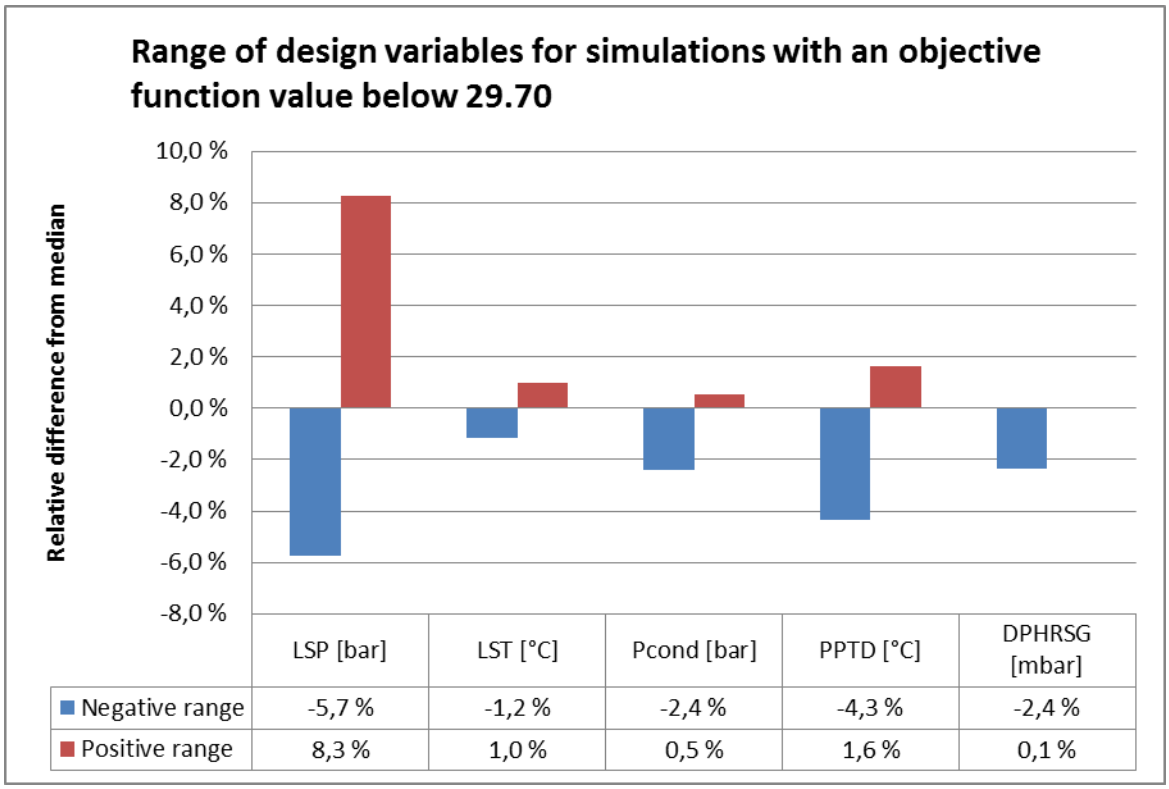

Figure 4.15 – Range of design variables in simulations with an objective function value below 29.70 relative to the median values of the same simulations.

4.6 Comparison to Project Work

Table 4.15 shows a comparison of the results from the optimized solution to what was obtained in the project work.

Table 4.15 – Comparison of optimized solution to the project work

	Project work	**Optimized solution**	**Absolute difference**	**Relative difference**
LSP [bar]	25	24,35	-0,65	-2,60 %
LST [°C]	470	488,69	18,69	4,00 %
Pcond [bar]	0,08	0,04856	-0,03	-39,30 %
PPTD [°C]	25	33,251	8,25	33,00 %
DPHRSG [mbar]	30	34,996	5	16,70 %
Objective function value	**30,4612**	**29,6682**	**-0,79**	**-2,60 %**
Weight HRSG [kg]	103 278	91 223	-12 055	-11,70 %
Weight ST [kg]	25 151	32 418	7 267	28,90 %
Weight ST generator [kg]	33 530	34 281	751	2,20 %
Weight condenser [kg]	12 388	15 932	3 544	28,60 %
ST power [kW]	11 182	11 536	354	3,20 %
GT power [kW]	31 970	31 884	-86	-0,30 %
Total weight components [kg]	**174 347**	**173 854**	**-492,59**	**-0,28 %**
Total power output [kW]	**43 152**	**43 420**	**268,02**	**0,62 %**

Looking at the design variables first, the relative differences are illustrated in Figure 4.16. For the LSP the difference is very minor, so it appears as if the project work choice was a quite decent. In the project work the LST of 470 °C was chosen due to the conspicuous increase in equipment weight for LSTs above 470 °C. The optimized solution however, has opted for a bit higher LST, allowing for more energy being transferred to the steam which then can potentially be extracted as power in the ST. The significantly lower condenser pressure in the optimized solution also seems to indicate a focus on extracting more power from the ST. In the project work the choice of the condenser pressure was essentially down to cutting as much weight as possible without losing too much performance, so there appears to be a fundamental difference in how to achieve the best possible solution. In the optimized solution the PPTD and DPHRSG are more or less cut from the same cloth; within the confines of the upper boundaries set they are essentially as large as possible without resulting in erroneous computations. It is quite easy to understand why; the performance loss associated with an increase in the DPHRSG is quite marginal compared to its potential weight savings and the same holds true for the PPTD. However, the potential efficiency loss attached to an increase in PPTD is larger than for the DPHRSG, also explaining why maximizing the DPHRSG seems to be prioritized in the optimized solutions. In the project work, using a higher PPTD and DPHRSG than what was eventually used was seriously entertained as a possibility. However, ultimately decided to stay well within the confines of the reviewed theory, and not use extreme outliers. With a bit more aggressive approach the values of the project work would have been a bit closer to that of the optimized solution.

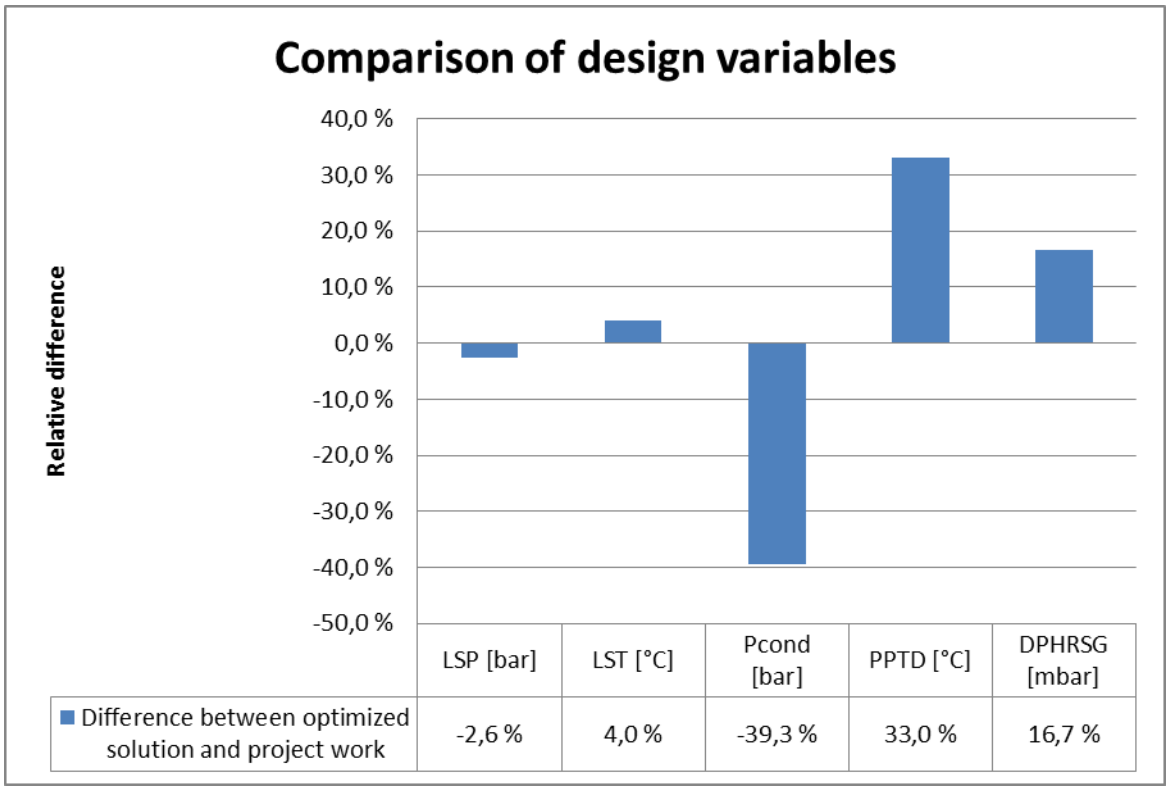

Figure 4.16 – Comparison of design variables between optimized solution and design obtained in project work.

The optimized solution was also compared to the project work at part-load, to check if the optimization had had any adverse effect on the system performance. As Table 4.16, Figure 4.17 and Figure 4.18 shows, that was not the case. If anything the relative part-load performance increased somewhat with the optimized solution, with the largest difference in net efficiency between the two solutions seemingly occurring at around 40 % and around 60 % of the GT load. In terms of absolute efficiency, the optimized solution was consistently better at all part-loads, though mainly due to having a better efficiency in the first place.

Table 4.16 – Part-load performance optimized solution compared to project work

Gas turbine load [%]	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Net efficiency - Project work [%]	42,24	44,77	45,35	47,07	48,67	49,89	50,48
Net efficiency - Optimized solution [%]	42,66	45,09	45,76	47,44	48,97	50,17	50,81
Difference	0,42	0,32	0,41	0,37	0,3	0,28	0,33

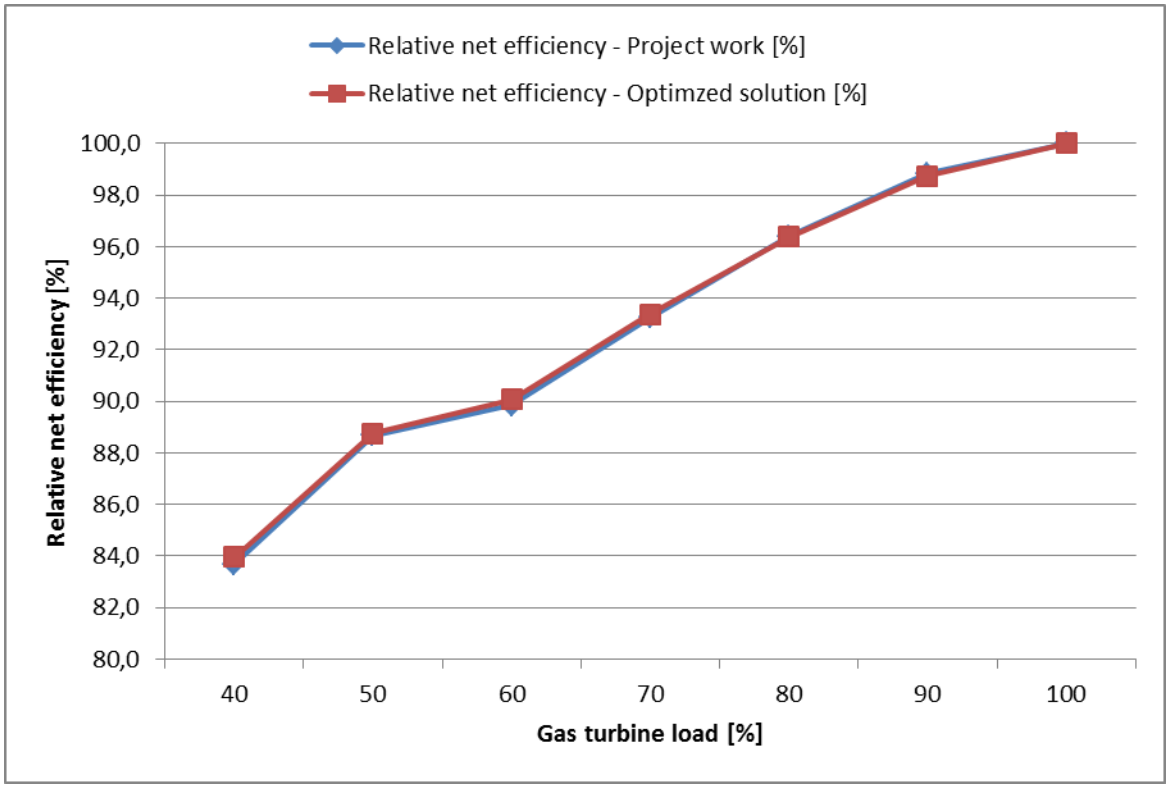

Figure 4.17 – Relative net efficiency of respectively the project work and the optimized solution at part-load.

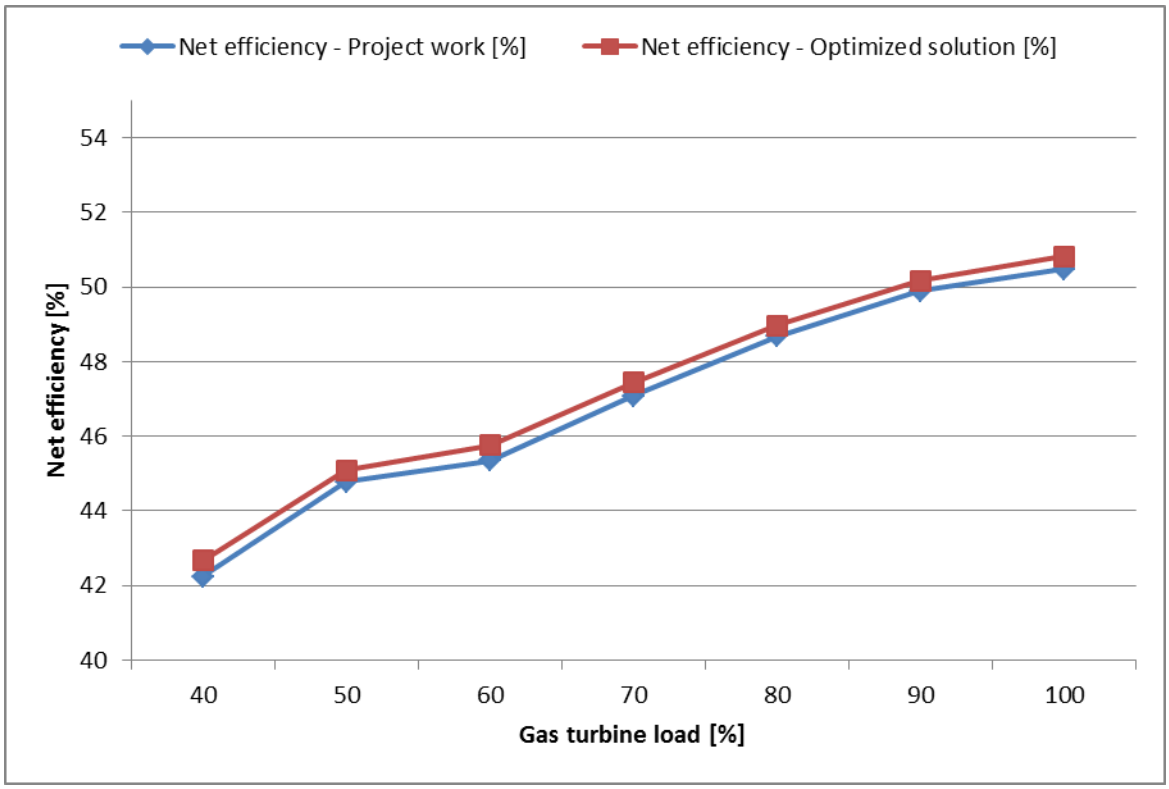

Figure 4.18 – Comparison of net efficiency of respectively the project work and the optimized solution at part-load.

So to summarize; the design differences between the optimized solution and project work suggests that the optimized solution appears to be more complete in its analysis, and not only focused on cutting weight like the project work was. This was perhaps illustrated the best by the apparent illogical choice of having a relatively low condenser pressure. In the project work it was assumed that the weight savings for the steam turbine and condenser associated with a higher condenser pressure would be advantageous in an offshore application. However, given the developed objective function, it turned out that in combination with the other optimized design parameters the increased power production with a lower condenser pressure outweighed the weight saving advantages of a higher condenser pressure. This apparent difference in philosophy is further emphasised when looking at the weight and power production of the different components. The optimized solution has shaved considerable weight from the HRSG system, in excess of 12 tons, and losing 86 kW from the GT as well as an unknown power loss in the ST. However, this power loss is easily recuperated with an increased power production from the ST as a consequence of the reduced condenser pressure and increased LST, resulting in a net power increase of 268 kW for the optimized solution. The weight cost of the increased power production can be seen in the larger ST and ST generator needed to handle the increased power, as well as the larger condenser needed for the reduced condenser pressure, totalling about 11.5 tons. As such, in total the optimized solution shaves off about half a ton of weight from the project work design, a weight reduction of 0.28 %, while increasing the power production by 268 kW, a power increase of 0.62 %. This may not sound like much, but when combined in

the objective function that also includes a bulk weight it adds up to a 2.60 % reduction in the value of the objective function, which can result in considerable savings when the total costs involved are as large as they are for offshore applications. In terms of part-load performance, the differences seem to be negligible.

4.7 Summary

As previously discussed, a simple cycle offshore GT has weight to power ratio of about 10. That is about three times better than the best weight to power ratio obtained for the steam cycle part of a CC in this thesis. However, as the steam cycle requires no additional fuel there are considerable fuel and CO_2 tax savings associated with using a CC, as well as CCs usually having a better part-load performance. Which solution is actually better is difficult to know without knowing the complete cost picture and performing a thorough life cycle analysis. What is known is that the CC solution developed in this thesis has a net plant efficiency of about 50.8 % at design point, which is about 33 % better than a simple cycle with the same GT. Additionally, the weight of the main components is considerably lower than for that of a high efficiency onshore CC, which has a weight to power ratio for just the main components of about 34, though its net plant efficiency is only a couple of percentage points worse. Consequently the CC optimized with respect to a weight to power ratio that includes a bulk weight for the necessary skid structures in addition to the main components looks like a quite good solution.

As previously alluded to, the first tested objective function only included the weight of the main components, which resulted in a partly conspicuously different design solution. It seemed to focus a lot more on saving weight; or rather, any weight changes had a much greater impact on the objective function than any changes in power output. The main difference with a different objective function was that the condenser pressure for the objective function only considering the weight of the main components was significantly higher at around 0.12 bar. Further, to recover some of the lost power output with an increased condenser pressure, the LSP was a bit higher at around 30 bar. The other design parameters were largely unchanged. Still, the optimized result for the objective function with a bulk weight was a relatively poor solution for an objective function without a bulk weight, only emphasising that the optimized solution is in fact only optimized for that particular objective function. The optimized result is only as good as the chosen objective function.

Generally speaking, it does seem very beneficial to optimize a plant design for combined cycles. However, as the optimized solution is very dependent on the choice of objective function, the choice of objective function is paramount for the optimization process. The optimized solution seems to depend a lot more on the choice of objective function than the choice of optimization method. For a given objective function all optimization methods were seemingly in the same ballpark in terms of the obtained solution, and without too much difference in simulation time. As such, if the problem at hand and what should be optimized

is well known, and an objective function can be developed to fit with that knowledge, there seems to be possible to achieve quite significant improvements in a design solution with a reasonably swift optimization search. If a more accurate representation of the true global optimum is sought after, adjustments can be made to either in the form of a different optimization method or by changes to the options structure of the optimization method used. As one would expect, the longer more thorough searches with more function evaluations are generally able to find the most improvement in the solution. The selection of initial point and boundaries seemed to be fairly insignificant for the performance of the pattern search with a MADSPositiveBasis2N search algorithm. However, a pattern search with a GA search algorithm seemed to perform the best when the boundaries were neither too relaxed nor too tight, though due to the stochastic nature of the GA there is some doubt about the validity of that claim. For the GA using a specified range for the initial population seems far more beneficial than using an initial point or narrower boundaries.

The best solution in this thesis was discovered by running a second pattern search using the final point of the best solution achieved so far as the starting point. Running one relatively quick optimization first, and then a second more thorough simulation using the solution from the first optimization as the starting point might be a very viable course of action.

5 Conclusion

This thesis has looked at optimizing combined cycles for offshore applications. This was done by developing an objective function thought to suitable address the presumed advantages and disadvantages of installing a combined cycle offshore, and then optimizing said objective function using MATLAB, GT PRO and Microsoft Excel. A small disclaimer is needed off the bat. The frequent crashes with certain settings and simulation methods were a bit of a nuisance, and consequently impeding the progress of some of the optimization attempts.

Through this thesis it has become apparent that the selection of objective function is of great importance. It must fit with what is desired of the solution, as the optimized solution will only be optimized for the scenario described by the objective function, and as such the optimized solution will only be as good as the selected objective function. In terms of choice of optimization method, the global optimization toolbox of MATLAB comes with several suitable alternatives, and all non-crashing algorithms were seemingly able to locate quite good solutions. There were no apparent differences in the design solutions between the various search methods; they were all in the immediate vicinity of each other.

In the end, the pattern search with a MADSPositiveBasis2N search algorithm seemed to be a good option for obtaining the best possible solution, while a pattern search with a genetic algorithm search seemed to be able to achieve a quite good solution in a fairly reasonable time, though struggling to locate the very best solutions. The genetic algorithm was also thoroughly investigated, and though it offered much room for improvement by adjustments to its options structure, it was neither able to obtain a very good solution nor particularly quick. As such it seemed more suitable as a backup method than as the method of preference.

In comparison to the design developed in the project work, there were noticeable improvements to be had in terms of power production and weight savings. While the project work had essentially focused on cutting as much weight as possible, the optimized solution was more balanced in its approach and able to consider several elements of the problem simultaneously. Overall, the optimized solution was 493 kg lighter and able to produce an additional 268 kW when compared to the project work, corresponding to a 2.6 % improvement in the objective function value. This may not sound like much, but the cumulative savings over the lifetime of an installation may become quite considerable. The optimized weight to power ratio for the steam cycle part of a combined cycle was about 3 times higher than that of a comparable gas turbine typically used offshore. However, as it requires no additional fuel it offers considerable savings in both CO_2 tax and fuel cost, perhaps making it the preferred solution when a complete life cycle analysis is performed.

In terms of changing the optimization parameters, the pattern search with a MADSPositiveBasis2N search algorithm was able to improve the final solution by essentially covering more of the region immediately surrounding the optimal solution, though naturally the increased number of points being searched increased the simulation time. Still, there

were fairly marginal differences between solutions for any pattern search with a search algorithm. Of note, for the pattern search with a MADSPositiveBasis2N search algorithm the selection of an initial point and boundaries had little influence on the final solution and total simulation time. The upper and lower boundaries set had seemingly little impact other than preventing GT PRO from having to try simulating processes that were clearly infeasible or entailing frequent erroneous computations.

Overall it appears to be quite advantageous to optimize the design of combined cycles for offshore oil and gas installations. Once a suitable objective function is established a quite good optimized solution can be realized in relatively short time. It does not appear to be necessary with many adjustments in the optimization parameters, though adjustments can be made if a better solution is aimed at.

6 Further Work

The optimization in this thesis was for a fixed HRSG geometry developed in the project work. However, there are other hardware configurations that could entail a better optimized solution. Potential further work could include optimizing the hardware geometry of the HRSG if the software allows it. As it was at the time of writing this thesis, changing the hardware geometry was not possible from the link with Microsoft Excel, and as such any optimization would have had to be done by manually changing the hardware parameters in GT PRO, much like what was done in the project work.

Other possible work includes attempting the optimization for different objective functions. Alternatives includes among others only optimizing with respect to weight or power, or perhaps developing a more accurate bulk weight. Most optimizations consists of minimizing the cost of something, so if the cost of each kg of installed weight as well as the running costs of fuel and CO_2 tax for a given power production could be developed, then that could be minimized over the expected lifetime of the installation and compared directly to the simple cycle gas turbines typically used offshore today.

7 References

Bolland, O., 2013. *Discussion of master thesis* [conversation] (Personal communication, 16 January 2013).

Edgar, T. F., Himmelblau, D., M, and Lasdon, L., S., 2001. *Optimization of chemical processes*. 2nd ed. Boston: McGraw-Hill.

Hellberg, A. 2006. *Experience of 29MW SGT-700 Gas Turbine in Power Generation Applications*. [pdf] Available at: <http://www.energy.siemens.com/fi/pool/hq/energy-topics/pdfs/en/gas-turbines-power-plants/8_Experience_of_29MW.pdf> [Accessed 12 April 2013].

Jaluria, Y., 2008. *Design and Optimization of Thermal Systems*. 2nd ed. Boca Raton, Florida: CRC Press.

Kloster, P. 1999. Energy Optimization on Offshore Installations with Emphasis on Offshore Combined Cycle Plants. In: Society of Petroleum Engineers Inc, *Offshore Europe Conference.* Aberdeen, Scotland 7-9 September 1999, paper no. SPE 56964.

Finansdepartementet, 2012. *Skatter, avgifter og toll 2013, Forslag til CO2-avgiftssatser for 2013* [online] Available at: <http://www.regjeringen.no/nb/dep/fin/dok/regpubl/prop/2012-2013/prop-1-ls-20122013/11/9/2.html?id=702766> [Accessed 07 June 2013].

MathWorld, 2013. *Convex Polyhedron.* [image online] Available at: <http://mathworld.wolfram.com/ConvexPolyhedron.html> [Accessed 12 March 2013].

Nord, L.O., 2013. *Discussion and counselling of master thesis* [conversation] (Personal communication, several meetings from 16 January 2013 to 28 May 2013).

Norwegian Petroleum Directorate, 2013. *Facts 2013 - The Norwegian Petroleum Sector.* [online] Available at: <http://npd.no/Global/Engelsk/3-Publications/Facts/Facts2013/FACTS_2013.pdf> [Accessed 07 June 2013].

Sletten, A., 2012. *Process simulation of combined cycles for offshore applications.*

Stoecker, W., F., 1980. *Design of Thermal Systems.* 2nd ed. New York: McGraw-Hill.

Thermoflow Inc, 2012a. *Products, Combined Cycle, GT PRO.* [online] Available at: <http://www.thermoflow.com/combinedcycle_GTP.html> [Accessed 13 December 2012].

Thermoflow Inc, 2012b. *Products, Combined Cycle, GT MASTER.* [online] Available at: <http://www.thermoflow.com/combinedcycle_GTM.html> [Accessed 13 December 2012].

Thermoflow Inc, 2012c. *Products, Combined Cycle, PEACE.* [online] Available at: <http://www.thermoflow.com/combinedcycle_PCE.html> [Accessed 13 December 2012].

The Engineering ToolBox, 2013. *Liquid – Densities*. [online] Available at: <http://www.engineeringtoolbox.com/liquids-densities-d_743.html> [Accessed 12 April 2013].

The MathWorks, Inc., 2012. *Global Optimization Toolbox User's Guide.* [pdf] Available through: MathWorks PDF Documentation Center. <http://www.mathworks.se/help/pdf_doc/gads/gads_tb.pdf> [Accessed 07 February 2013].

Tollvesenet, 2013. *Excise Duty on Emissions of NOx 2013.* [online] Available at: <http://www.toll.no/upload/aarsrundskriv/Engelske/2013%20Emissions%20of%20NOx.pdf> [Accessed 07 June 2013].

Appendix A – Bulk Weight Estimation Motors and Pumps

Estimated weight of motors and pumps in plant, extracted from PEACE.

	Number of units	Weight [kg]
Feedwater Pump	3	
Pumps		338
Motor		209
		1641
Condensate Forwarding Pump	2	
Pumps		148
Motor		66
		428
Condenser C.W. Pump (P4)	2	
Pumps		1380
Motor		
		2760
Condenser Vacuum Pump (P7)	2	
Pumps		1380
Motor		
		2760
Treated Water Pump (P11)	1	
Pumps		81
Motor		25
		106
Demin Water Pump (P23)	2	
Pumps		56
Motor		15
		142
Raw Water Pump 1 (P26)	1	
Pumps		71
Motor		21
		92
Raw Water Pump 2 (P27)	1	
Pumps		71
Motor		21
		92
Aux Cooling Water Pump (closed loop) (P10)	2	
Pumps		92
Motor		33
		250
Diesel Fire Pump (P12)	1	
Pumps		1440
Motor		

		1440
Air Compressor	2	
Pumps		889
Motor		
		1778
Total weight motors and pumps		**11489**

Appendix B - MATLAB Code

Objective function to be optimized

```
function WP = WeightToPower(x) % Weight to power objective function to be
optimized

global e

% Writes the input value for the simulation
ePres = e.Activesheet.get('Range', 'E8'); % Live steam pressure
ePres.Value = x(1);
eTemp = e.Activesheet.get('Range', 'E9'); % Live steam temperature
eTemp.Value = x(2);
ePCond = e.Activesheet.get('Range', 'E10'); % Condenser pressure
ePCond.Value = x(3);
eDT = e.Activesheet.get('Range', 'E11'); % Minimum pinch point temperature
difference
eDT.Value = x(4);
ePHRSG = e.Activesheet.get('Range', 'E12'); % HRSG draft loss
ePHRSG.Value = x(5);

BWeight = 150000; % Estimation of additional bulk weight from HRSG and ST
skid, including skid structure, water tanks, pumps, makeup water system++

e.ExecuteExcel4Macro('!Therm()'); % Executes Excel Macro 'Therm'
pause(2) % pause time in seconds (estimated time to complete one Thermoflow
run)

% Reads the cell with the Thermoflow Computation Message (OK, Messages,
Failed!)
eFlag = e.Activesheet.get('Range', 'E6');
flag = eFlag.Value;

% Generates weight-to-power ratio
 if strcmp(flag,'OK')
eWeightHR = e.Activesheet.get('Range', 'E15'); % Wet weight HRSG
eWeightST = e.Activesheet.get('Range', 'E17'); % Weight steam turbine
eWeightG = e.Activesheet.get('Range', 'E18'); % Weight generator steam
turbine
eWeightC = e.Activesheet.get('Range', 'E20'); % Wet weight condenser
eSTPower = e.Activesheet.get('Range', 'E16'); % Power from steam turbine
eGTPower = e.Activesheet.get('Range', 'E19'); % Power from gas turbine

GTrefP = 32504; % Power output from GT when run as a simple cycle with
exhaust losses of 4.99 mbar
Weight =
eWeightHR.Value+eWeightST.Value+eWeightG.Value+eWeightC.Value+BWeight; %
Weight of all components in steam cycle
STPower = eSTPower.Value;
GTPower = eGTPower.Value;

Power = STPower - (GTrefP - GTPower); % Additional power from adding a
steam cycle
WP = Weight/Power; % The weight to power objective function to return
elseif strcmp(flag,'Messages') % If there are any warning messages, WP is
set to a high value so the point will be "ignored"
```

```
    WP = 50; % The weight to power objective function to return when
computation is erroneous
 else
     WP = INF; % If the computation completely fails, WP is set to
infinity, and the optimization stops
 end

end
```

Pattern search algorithm

```
% Direct Search using pattern search with the possibility of including the
following search methods: GPSPositiveBasis2N, GPSPositiveBasisNp1,
GSSPositiveBasis2N, GSSPositiveBasisNp1, MADSPositiveBasis2N (current
search method being used), MADSPositiveBasisNp1, searchga(GA search),
searchlhs (Latin hypercube search) and searchneldermead(Nelder-Mead
search).
lb = [15 400 0.0234 8 15]; % Lower boundary for x, LSP, LST, PCond, PPTD,
DP HRSG
ub = [35 510 0.14 35 35]; % Upper boundary for x, LSP, LST, PCond, PPTD, DP
HRSG
x0 = [25 470 0.08 25 30]; % Initial guess of best point, values from
project work
tic; % Start clock
tStart = tic;
% Creates options structure for altering the default settings of the
% pattern search algorithm
options =
psoptimset('SearchMethod','MADSPositiveBasis2N','PollMethod','GPSPositiveBa
sis2N','CompletePoll','off','MeshContraction',0.8,'MeshExpansion',3,'MaxIte
r',2000,'MaxFunEvals',40000,'display','iter','PlotFcns',@psplotbestf);
[x fval] = patternsearch(@WeightToPower,x0,[],[],[],[],lb,ub,[],options); %
Returns the value of the objective function at the solution x
tElapsed = toc(tStart); % Returns the time elapsed in seconds
```

Genetic algorithm

```
% Genetic Algorithm (GA)
lb = [15 400 0.0234 8 15]; % Lower boundary for x, LSP, LST, PCond, PPTD,
DP HRSG
ub = [35 510 0.14 35 35]; % Upper boundary for x, LSP, LST, PCond, PPTD, DP
HRSG
nvars = length(lb); % Number of design variables
tic; % Start clock
tStart = tic;
% Creates options structure for altering the default settings of the GA
% PopInitRange is chosen based on the immediate region surrounding the
project work values
options =
gaoptimset('populationsize',100,'generations',100,'StallGenLimit',20,'TolFu
n',0,'EliteCount',5,'PopInitRange',[20 440 0.04 20 25;30 500 0.12 33.3
35],'display','iter','PlotFcns',@gaplotbestf);
[x,fval] = ga(@WeightToPower,nvars,[],[],[],[],lb,ub,[],options); % Returns
the value of the objective function at the solution x
tElapsed = toc(tStart); % Returns the time elapsed in seconds
```

Simulated annealing, frequent crashes so not really used in thesis

```
% Simulated annealing
lb = [15 400 0.0234 8 15]; % Lower boundary for x, LSP, LST, PCond, PPTD,
DP HRSG
ub = [35 510 0.14 35 35]; % Upper boundary for x, LSP, LST, PCond, PPTD, DP
HRSG
x0 = [25 470 0.08 25 30]; % Initial guess, values from project work
tic; % Start clock
tStart = tic;
options =
saoptimset('display','iter','PlotFcns',@saplotbestf,'PlotFcns',@saplotf);
[x fval] = simulannealbnd(@WeightToPower,x0,lb,ub,options); % Returns the
value of the objective function at the solution x
tElapsed = toc(tStart); % Returns the time elapsed in seconds
```