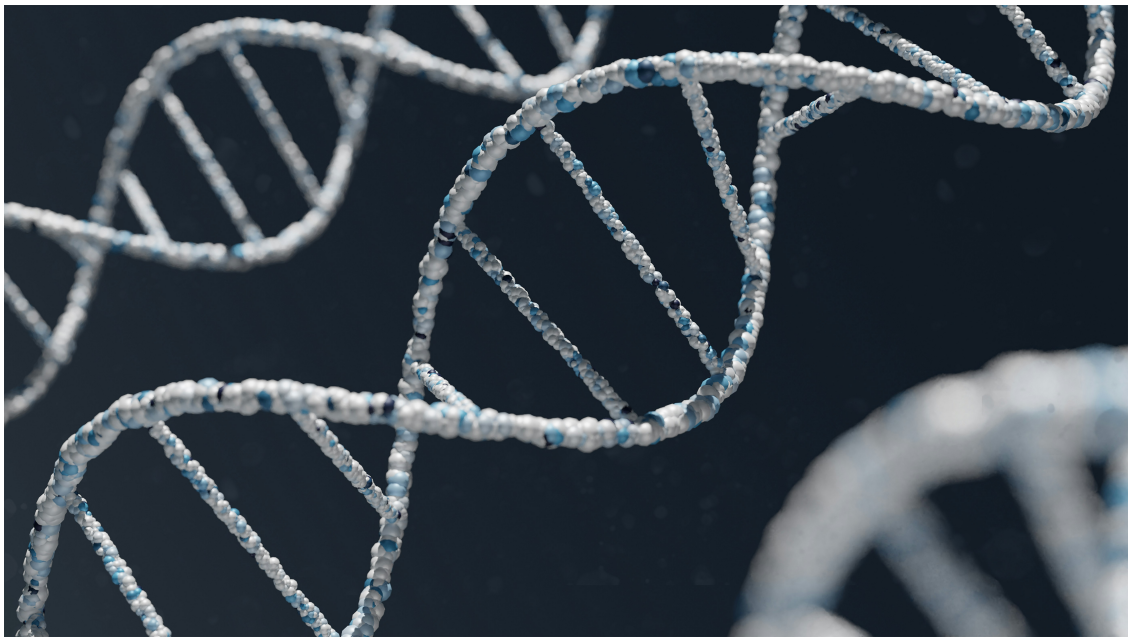


Optimizing Microgrid Energy Scheduling

Comparing performance of Genetic algorithms and Mixed Integer Linear Programming

Master's thesis in Complex Adaptive Systems

PONTUS STRANDVIK

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Optimizing Microgrid Energy Scheduling

Comparing performance of Genetic algorithms and Mixed Integer Linear Programming

PONTUS STRANDVIK

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2026

Optimizing Microgrid energy scheduling
Comparing performance of Genetic algorithms and Mixed Integer Linear Programming
PONTUS STRANDVIK

Supervisor: Leon Sütfeld, RISE
Examiner: Malin Rau, Department of Mathematics

Master's Thesis 2026
Department of Mathematics
Division of Applied Mathematics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 70 775 9660

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Optimizing Microgrid Energy Scheduling
Comparing performance of Genetic algorithms and Mixed Integer Linear Programing
PONTUS STRANDVIK
Department of Physics
Chalmers University of Technology

Abstract

This thesis deals with the comparison of Genetic algorithms and Mixed Integer Linear Programming on the microgrid energy scheduling problem. The focus lies on genetic algorithms and improvements to problems experienced by the GA with MILP being used as a baseline for comparison. The GA and MILP are being compared on the unit commitment / economic load dispatch problem within a microgrid setting. The objective is to investigate under which circumstances the individual optimizers are preferable. In the GA approach constraints were handled using penalty terms added in the fitness function encoding physical constraints such as battery storage limits. Additional penalties were used to enforce objectives such as temperature ranges and binary statuses of components. In the MILP approach constraints were handled in the model creation with objectives encoded as terms in the objective function scaled by individual weight factors. The research was carried out on a simulation tool where two scenarios were modeled. One scenario represents a microgrid in a remote setting without external grid connection where energy consumption minimization was of importance to prolong the useful life of components. The second scenario represents a grid connected microgrid where precise temperature management and energy cost minimization through strategic energy market interactions was the main objective. The results suggest that the MILP outperforms the GA on most of the problems considered with the exception of certain cases of uncertainty in the renewable energy generation. Results also show that improvements could be made to the genetic algorithm by seeding the initial population. Furthermore, the gentic algorithm performance and computation time improved on the mountain hut scenario when a timestep of 8-16 hours was used. This result is unexpected as a lower temporal resolution would make the problem harder to handle, however in the case of maximizing up time of components a larger timestep duration seems preferable.

Keywords: optimization, genetic algorithms, MILP, microgrid

Acknowledgements

I would like to thank my supervisor Leon Sütfeld for his support and guidance during the master's thesis process. I would also like to thank my thesis partner Arnaud Sandell for being a excellent discussing partner and for consistently contributing with valuable ideas. Finally i would like to thank my examiner Malin Rau for her support on practical matters of the thesis.

Pontus Strandvik, Gothenburg, 06 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

UC	Unit commitment
ED	Economic dispatch
MILP	Mixed-Integer Linear Programming
MG	Microgrid
PV	Photovoltaic
EV	Electric vehicle
GA	Genetic algorithm
SOC	State of charge

Contents

List of Figures

List of Tables

1

Introduction

1.1 Introduction

A microgrid (MG) is a flexible alternative to the main electricity provider that enables support where traditional energy providers encounter problems, such as in remote or off grid locations where grid infrastructure is absent or economical unfeasible to build. Energy management systems are needed to coordinate the scheduling and maintain a balance between supply and demand. Poor scheduling can lead to wasted energy, unnecessary emissions and higher costs, both in energy expenditure and in replacement costs of devices that might get an increased replacement rate with suboptimal management [36].

The optimization of MGs represents a multidimensional challenge where operators must simultaneously minimize energy costs, manage potential storage degradation, satisfy load demand, and respect physical operating constraints, while accounting for the stochastic output of renewable energy sources. These competing objectives produce a optimization problem that is difficult to solve exactly at scale. Mixed Integer Linear Programming (MILP) offers a exact approach to this problem class but depending on the problem size it can carry a significant computational cost. Genetic Algorithms (GA) offer a flexible alternative with the added benefit of being able to handle nonlinear as well as linear problem formulations. To compare the both approaches, this thesis adopts a power systems modeling simulation in which the optimizers will operate.

Optimizing a UC/ED scheduling problem is complex and constraints between devices such as SOC limits and fuel availability must be satisfied simultaneously creating a complex solution landscape. The presence of renewable energy sources introduces uncertainty in available energy as the optimizer relies on predicted wind and solar data that may deviate from actual conditions. The two optimizing frameworks being compared in this thesis is GA and MILP. These differ in the way that MILP offer a deterministic solution to the optimization problem with a guarantee of optimality. It requires a linear model of the system, meaning that if there exists nonlinear behavior in the system it must be must be approximated. GAs on the other hand offer a flexible alternative without any constraint on model linearity, searching the solution space through population based evolution. The drawback is that GA offer no guarantee of optimality. GA has a linear runtime complexity in the number and size of chromosomes evaluated as long as the fitness function evaluation also is linear, while a very large binary decision state space can effect computational

performance of the MILP [22].
The two scenarios investigated in this thesis is a mountain hut scenario which represents a system operating without connection to the main grid. It is designed to support telecommunications in remote regions where connectivity is otherwise absent. A defining feature of this scenario is a semi annual refueling event which motivates a long planning horizon where the optimizer must plan the energy use across months in advance to account for fuel availability and maximize the up time of the components. This scenario represents the situation where a MG must make the most of the energy that it has available. The second scenario represents the grid connected farm MG where access to the main grid changes the optimization objective from resource conservation to cost minimization through strategic interaction with the energy market. The farm scenario includes heat as a second energy carrier increasing the complexity of the simulation and scheduling process of the optimizer relative to the mountain hut scenario.

1.2 Objective

The objective of this thesis is to characterize the conditions under which either Genetic algorithms (GA) or Mixed Integer Linear programming (MILP) optimization is preferable for two simulated scenarios. This is to be investigated via applying GA and MILP to a Unit commitment (UC) / Economic load dispatch (ED) scheduling problem within a microgrid setting. Uncertainty is included in energy production and in load demand. The optimizers are evaluated and compared against a ground truth metric which represent important statistics about the solution quality. A simulator is also developed as part of the thesis, which generates scenarios on which the optimization algorithms are evaluated.

1.3 Scope

The comparison of optimizers is limited to GA and MILP. Two energy system scenarios are considered: a scenario that involves loads, storage, and energy production using electricity and fuel as energy carriers, and a larger scenario that includes heat as an energy carrier and a connection to the electricity grid. The uncertainty in renewable production and grid spot prices is derived from historical data and historical forecast data, while the uncertainty in demand is modeled statistically. The optimizers are evaluated in simulation only and will not be validated on physical hardware. Energy sources beyond those described in the two scenarios are not considered. The planning horizon ranges from 24 hours to 5000 hours ahead, with a minimum resolution of one hour for optimizer decisions. Component models are linear for storage and load, whereas nonlinear models are used for renewable energy production.

2

Literature review

2.1 GA for microgrid scheduling

Genetic algorithms are a class of population-based metaheuristic optimization methods inspired by the principles of biological evolution. Their use is well established in microgrid scheduling and are used for complex optimization problems with nonlinear features. In the context of operational scheduling GAs have been shown to produce competitive solutions for UC and ED in both islanded and grid-connected microgrids [11]. The optimization of a microgrid is usually multiobjective where objectives are competing against each other where the improvement of one objective leads to a deterioration of another. Such objectives can typically be cost minimization and emission reduction for a microgrid setting. Many papers use day ahead scheduling with a timestep of 60 minutes with energy balance constraints between components being upheld for every timestep [26][33]. In many papers uncertainties are handled using simulation based optimization, robust or stochastic optimization [6].

In GA optimization every schedule is encoded as a chromosome and choosing an encoding structure should be done with regard to the given optimization problem, as no encoding structure is better for every problem [34]. An example of a schedule encoding is given in a study by Witharama et al optimizing battery dispatch each battery schedule is encoded as a triplet of decisions relating to the battery [33], where each gene represents decisions of the battery. Other encodings that are used are variable length encoding where scheduling decisions are varied based on predetermined settings [12].

The fitness function is commonly a reflection of the objective of the optimization, often through weighted sum approach or through Pareto optimization [28].

Binary variables encodes on/off decisions for each unit at each timestep and continuous variables encodes power output of a device per unit and timestep. A population of chromosomes are initialized, evaluated against a fitness function and iteratively evolved through selection, crossover, and mutation until a termination criteria is met. The result is a hopefully near optimal schedule produced without the need for linearizing the problem which allows the GA to work with nonlinear component models [24].

2.1.1 GA parameters

The choice of population size depends on the problem size and typically ranges from 50 to 150 individuals [28] [24] [32]. The computational time of the GA increases with population size and to reduce computational time more advanced GA approaches have been proposed. One such method deals with premature convergence where a large injection of new genetic material is made into the population via a cataclysmic restart, where a large portion of the population is mutated only a few elite individuals are kept unchanged [37]. Constraints are usually handled via large negative rewards added to the fitness function, filtering out individuals that violate constraints or via a repair function altering the solutions that violate the constraints [33], [3].

2.1.2 Hyperparameter tuning

GA performance is sensitive to the choice of hyperparameters such as population size and mutation rate [34]. Some papers report tuning strategies such as [28] and point out that the choice strongly effect the outcome. Some papers utilize grid search across hyperparameters to find the best ones for the current problem [31]. Other papers propose more advanced methods such as adaptive mutation rates encoded into the chromosome, reducing the number of parameters to manually tune [30]. A third option that is presented in the literature is a meta optimization where a GA optimization is done on top of the optimization that encodes the hyperparameters of the GA as genes in the chromosome [5]. This is however a slow process as one fitness evaluation becomes one complete run on the original optimization. General values that are known to work well on many problems can also be used with the downside that it might be a suboptimal parameter value creating slower convergence speed or premature convergence.

2.1.3 Limitations

The Genetic algorithms have a known problem of premature convergence where the GA can become trapped in a local optima. In the context of scheduling each candidate solution represents a schedule proposed by the GA. Because the UC/ED scheduling problem exhibit issues with solution spaces growing rapidly, convergence is a key concern when developing a GA. Hybridization approaches are popular where the GA is combined with other heuristic or deterministic algorithms to utilize the good properties of the separate approaches [37][24]. One example is given in [29], which proposes a fast genetic algorithm optimization for the UC/ED scheduling problem. It is done by enforcing minimum up time in the initialization of the chromosome, by using clustering of similar components to reduce the size of the chromosome, using a priority list seeding algorithm that seeds the initial population with good solutions and also by using an alternate way to do mutation where minimum up time constraints are not violated by the mutation. They also propose enforcing fewer heavy computations by only evaluating computationally heavy parts if some of the most important constraint were met for the proposed solution. These are examples of how the GA has been combined with hybrid methods and additional ideas to decrease computation time.

Population diversity is another important topic in the GA optimization. The GA can suffer from premature convergence if the population diversity shrinks to early in the search process. Several methods to address this problem has been proposed. One such method is the migration ring (MigRing) approach which introduces islands with sub-populations that evolve separately and periodically exchange individuals to preserve diversity across the total population, increasing convergence time and solution quality [9]. Another method keeps track of a crowding distance metric which promotes individuals that are not clustered together [20].

2.2 Microgrid scheduling

The scheduling problem in a microgrid is the problem of finding a schedule for a given combination of units in a system while at the same time satisfying a set of operational constraints [21]. It is important to consider because poor scheduling decisions leads to wasted fuel, unnecessary emissions, higher electricity prices and in worst case, potential blackouts [13].

The problem is difficult for a couple of reasons. Firstly, the binary commitment decisions create a solution space that grows exponentially with the number of controllable units, making exhaustive search intractable beyond small system sizes. Second, the heterogeneous nature of microgrid components introduces a number of coupled constraints such as start up costs, storage state-of-charge limits, and energy balance equations that must be satisfied simultaneously. Third, the configuration of a microgrid can vary greatly which require an optimization algorithm that can handle a large number of varying cases of variable encoding, constraints and components. Finding the global optimum is generally impractical for real-sized systems, and some parts of the literature has focused on algorithms that reliably find near-optimal solutions within acceptable computation times [11]. Both exact methods such as MILP and metaheuristic methods such as genetic algorithms have been applied to this problem, and the choice of method involves a trade-off between solution quality and computational tractability that becomes increasingly pronounced as system complexity grows.

2.3 MILP for microgrid scheduling

Mixed-integer linear programming (MILP) is a mathematical optimization framework in which some decision variables are constrained to integer values. This structure maps naturally onto the UC/ED scheduling problem, where unit commitment decisions are binary and dispatch quantities are continuous. A MILP formulation of the scheduling problem consists of an objective function typically addressed to limiting costs and a set of linear constraints encoding the physical and operational limits of each component. These constraints include energy balance equations, charge and discharge exclusivity, power output limits and storage state-of-charge bounds. The resulting model is solved using solvers such as Gurobi or HiGHs, which systematically explore the solution space and return a provably optimal solution with respect

a chosen optimality gap and to the linearized model. The principal strength of MILP is its optimality guarantee. Because branch-and-cut exhaustively explores the feasible space, the solution returned is guaranteed to be globally optimal or within a specified gap of the global optimum. This determinism is a significant advantage in scheduling applications, and it distinguishes MILP from metaheuristic methods which offer no such guarantee [23]. A further advantage is flexibility in constraint specification where additional operational requirements can be incorporated as linear constraints without restructuring the solution method. The principal weakness of MILP is computational scalability. The unit commitment problem is NP-hard, meaning solutions cannot be found with a polynomial time algorithm. In practice, the computational bottleneck arises within the branch-and-cut process, where large linear relaxations must be solved repeatedly and conventional solver methods become expensive as the number of variables and constraints increases [22]. For the small systems considered in this thesis this scalability limitation is unlikely to be binding. However, as the number of controllable units and time steps increases, MILP tractability degrades, motivating interest in alternative methods. A further limitation relevant to microgrid applications is that MILP requires all constraints and the objective function to be linear. Many component models such as battery degradation curves, heat pump coefficient of performance and fuel cell efficiency at partial load are inherently nonlinear. Incorporating these accurately requires either linearization through piecewise approximations, which increases model complexity, or linearization of the used components models accepting that the optimized solution is optimal with respect to an approximated model rather than the true physical system [19].

3
Technical background

This chapter provide a background to the GA and the MILP and attempts to answer the question why a comparison of optimizers is needed and why it is important.

3.1 Mixed Integer Linear programming

MILP is a mathematical optimization framework that is a variant of Linear programming where some of the variables are restricted to be integer. This problem structure maps on to the scheduling problem in microgrids because it can represent power outputs as continuous values and on/off states as binary variables.

3.1.1 Problem formulation

A MILP problem formulation consists of a minimization of a linear objective function

$$\min \quad c^T x + d^T y$$

subject to constraints

$$Ax + By \leq b$$

with

$$x \in \mathbb{R}^n, y \in \{0,1\}^m.$$

In the context of this thesis x represents the continuous power output levels and y represents the binary commitment decisions. c and d are cost coefficient vectors and A and B represent the constraint matrices that map the decision variables to the constraint expressions and b is the right hand side vector defining the constraint bounds.

3.1.2 Branch and bound

The branch and bound approach was first described by Ailsa Land and Alison Doig and is used internally within the MILP framework [17]. The problem is first solved without the integer constraint with the simplex algorithm creating an optimal solution to the LP relaxation of the problem. The LP relaxed solution has more freedom than the integer problem, so the solution provides a lower bound to the optimization problem with integer constraints. If this solution satisfies all integrality constraints the problem is solved. If not, cutting planes are added to tighten the relaxation. The problem is then branched by selecting a non integer variable and creating two

subproblems, one where the variable is rounded up to the nearest integer and one where it is rounded down. The LP relaxation is solved at each node in the search tree. If a feasible solution is found that satisfies all integer constraints and the value of the solution is the most optimal found so far, it becomes the new incumbent solution, providing an upper bound on the feasible solutions. Nodes whose LP relaxation value is worse than the current incumbent are pruned and discarded, since no integer solution in that subtree can improve the incumbent. The search continue until the relative gap between the incumbent and the best remaining lower bound falls within a predefined tolerance.

3.2 Genetic algorithms

Genetic algorithms (GAs) belong to a family of metaheuristic optimization algorithms called evolutionary algorithms (EAs) [14]. The aim of the EA is to optimize a fitness function by mimicking the process of natural selection. The GA is a flexible optimization method that can be used to optimize any sort of problem formulation where the outcome can be quantified [14]. In the GA optimization process a population of solution vectors are created called chromosomes. These chromosomes encode the schedule of the components and receive a fitness score after evaluation based on the goodness of the proposed schedule. The fitness function is evaluated iteratively by the chromosomes in an effort to explore the solution space. The tradeoff between exploring the solution space and exploiting solutions in the current population is central in GA optimization and is determined largely by the values of the operator hyperparameters [14]. A large exploration is beneficial early on to explore the search space, followed by exploitation later in the optimization to refine the found solution [7]. The central operators in GA that govern the exploration/exploitation tradeoff is crossover, mutation and selection.

3.2.1 The GA loop

The process of GA optimization is described by the evolutionary algorithm workflow shown in Figure 4.1. The process starts by initialization of a population with a chosen encoding structure. The chromosomes are then evaluated based on the fitness function, put through selection, mutation and crossover to create the next generation. When a termination criteria has been met the algorithm terminates.

3.3 Microgrid structure

A microgrid is, as previously noted, a flexible system consisting of varying heterogeneous components. A problem arises when choosing an optimizer because of the varying possible microgrid configurations. This creates a need for an optimizer to be able to handle varying encoding structure and components. For a optimizer to be well suited for microgrid scheduling it must be evaluated on optimality, variable encoding possibilities and error of linear approximations. Other parameters that must be taken into consideration are the sensitivity to noise in the data since the

optimization process of a microgrid schedule will likely experience noise affecting the decisions.

3.3.1 Battery degradation

The following battery degradation model demonstrates that the selection of an appropriate optimization algorithm is scenario-dependent, particularly when closer approximation of component behavior is desired.

A objective that might give rise to a nonlinear objective is battery degradation modeling. Most microgrids utilize a battery to store energy and a simplified degradation model can represent the degradation of the battery as a function of the depth of discharge given by Equation B.1.1. Depth of discharge (DOD) of a battery refers to the fraction of a battery capacity which is removed at a specific discharge cycle. If 40 % of the total battery capacity is discharged then the DOD is 40%. There is a correlation between the depth of discharge and the cycle life of a battery where large DODs correlate with decreased cycle life [18]. Figure 3.1 shows how the depth of discharge relates to cycle life.

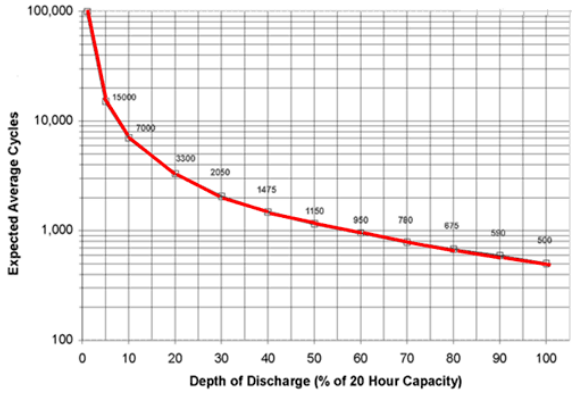


Figure 3.1: Cycle life of a battery depending on the depth of discharge

In an effort to minimize battery degradation, it is advised to minimize deep discharges [18].

For some microgrid configurations, battery degradation might not contribute significantly to the cost, in which case approximated modeling can be used. Such a model could be to minimize battery use which reduces battery degradation. However, for other configurations, it might represent a large portion of the cost and be competing with other objectives, in which case a more accurate representation would be beneficial. This illustrates that different microgrid configurations could have need of different optimization algorithms that work well for the specific use case.

3.4 Encodings

Both the MILP and the GA can represent binary, integer and real coded variables in the optimization process. The GA optimization process shows no benefit or downside of using either encoding structure but rather that the GA performs best with variables encoded to fit the structure of the problem [10]. The MILP however performs best when variables are continuous as no integrality constraints exists [25]. The problem can therefore be solved fast and precise with the simplex algorithm. The GA can handle constraints using penalties, repair or feasibility first rules [4]. The choice again depends on the problem where using penalties and feasibility first rule allows for infeasible solutions to exist in the population pool. The MILP handles constraints in the model creation, setting up limits for which the variables must be constrained within and otherwise determines the modeling problem as infeasible. The difference between GA and MILP is that infeasible solutions can be allowed to exist in the GA optimization creating the possibility for infeasible solutions to be returned by the GA. Both the GA and MILP can handle multiobjective optimization, most commonly through a weighted sum of objectives.

3.5 Robustness

Robustness refers to how consistently and reliably each method produce good solutions given varying conditions of noisy input data, different starting points and parameter choices.

The GA is heavily dependent on initial population as this represents the initial diversity of the population and a poorly seeded initial population can bias the search to a suboptimal region [16]. The population can be initialized in different ways but it is common to use random initialization which means that some initializations will produce better results and some worse. The MILP on the other hand is largely insensitive to initialization as it uses the branch and bound method which methodically searches through the search space. Using a warm start for the MILP can be beneficial for computational performance but the final solution is deterministic [1].

The GA has many hyperparameters to tune and performance is sensitive to these choices [14]. The MILP has few hyperparameters but can use an optimality gap tolerance specifying the closeness to the best solution but is generally not hard to use. The GA also has a known weakness of premature convergence [8]. If the solution space is large and the exploration is too constrained and the initial populations is biased to a search region, the population can converge at a local optimum. The MILP does not suffer from the same problem as it provably finds the best solution given a tolerance gap or returns that the problem is infeasable [25].

The GA is a more flexible alternative but require careful tuning of hyperparameters and multiple runs on a problem where the MILP offer a deterministic solution with few hyperparameters.

3.6 Scalability

Scalability refers to the ability of the optimizing algorithm to handle large problem sizes. This becomes important as some microgrid configurations will need to solve an optimization with large number of variables. The GA scales linearly with chromosome length and population size, given a linear fitness evaluation step [14]. The MILP however, has a known problem with combinatorial explosion where if the number of variables become too large the computation time quickly become intractable making microgrids with a large number of decision variables more suited for optimization with genetic algorithms [25], [8].

3.7 Optimality

The MILP has a guarantee of finding the optimal solution, given a linear model [23]. It also produces a relative optimality gap, specifying the distance between the current best solution and the best solution found for the relaxed problem. The GA offer no such guarantee creating difficulty in knowing the optimality of the found solution.

4
Methods

4.1 Simulation overview

In this thesis, a flexible simulation setup was implemented used to evaluate the optimizer performance. The simulation handles input/output calculations by decoding schedules produced by the optimizers and implements a battery SOC management mechanism. To account for varying needs in temporal resolution of the optimizer the simulation could be initialized with either a fixed or a varying length timestep allowing different parts of the simulation to be optimized with different temporal resolutions. A rolling horizon optimization was also implemented, where a planning horizon combined with a control horizon allowed for the use of piecewise optimization of the entire simulated period. For the GA the simulation was also used as part of the fitness calculation where every chromosome was simulated and the resulting system state was penalized. The optimizers were initialized with data containing information about the weather and grid spot prices. To investigate optimizer performance under forecast uncertainty the optimizers were initialized with data based on a forecasted version of the weather, optimizing the decisions based on predicted conditions. The optimized schedules were then evaluated on the ground truth weather data to evaluate true performance.

4.1.1 Battery SOC handling

A battery SOC handling system was applied in the simulation based on the concept of hysteresis load shedding. If a certain gene in a chromosome would have put the battery SOC below 10 % then the battery sheds all load until the SOC have risen above 20%. This was done to help guide the GA to find solutions where the battery does not flicker around the cut off threshold. To remove unfeasible values where the battery had a SOC above 100 % generating components were shut of if it would have increased the SOC above 100 %. The same mechanism was applied to the schedule proposed by the MILP after the optimization was done.

4.1.2 Uncertainty modeling

The uncertainty was modeled by using past predicted weather forecasts. The forecast data is generated so that the value on any given timestep is the forecast from a number of days ago where the number of days are specified in the forecast lag. This was done to model a general uncertainty in the weather data to emulate varying forecast accuracy levels.

4.1.3 Time handling

The time in the simulation had a flexible setup and several parameters could be varied to create different time structures.

Simulation timestep

The simulation timestep used was a zero order hold update rule described as

$$SOC[t] = SOC[t-1] + (P_{gen}[t-1] - P_{load}[t-1]) \cdot \Delta t$$

Timestep duration The timestep duration defines at what intervals the optimizer could make decisions and was a multiplier of the minimal timestep of 1 hour. The varying timestep option enabled the optimizer to make detailed decisions at some interval in the optimization while making decisions using a more coarse timestep in other intervals allowing for more flexibility in optimizer decisions.

4.1.4 Components models

Below follows a description of the models used to describe the components used in the simulation.

Battery model

The modeling of the battery was chosen to be a discrete time linear model implemented as follows:

$$\mathrm{SOC}_b(t) = \mathrm{SOC}_b(t-1)\Big(1 - \sigma_b(t)\Big) + \frac{r_s(t-1)}{E_s}\,\eta_s^{\mathrm{ch}} - \frac{r_s(t-1)}{E_s\,\eta_s^{\mathrm{dis}}}$$

Where t is the timestep, σ_b is the self discharge rate of the battery, r_b is the output of the battery, η^{ch} and η^{dis} are the charging and discharging efficiencies, and E_b is the capacity of the battery. This model assumes a linear relationship in charging and discharging the battery.

Battery degradation model

Battery degradation is calculated by using a rain flow counting approach combined with a logarithmic depth of discharge stress function and linear damage accumulation. The rain flow counting algorithm extracts cycles of varying depth from the SOC time series. Each cycle is binned together with other cycles that are approximately equally large. The battery degradation was calculated as an accumulation of the fraction of the battery life that was consumed

$$\text{Battery degradation} = \sum_{i}^{I} \frac{\gamma_i}{L(\xi_i)}$$

Where γ_i is the number of discharges at a specific DOD given by the rain flow counting, ξ_i is the DOD value and I is the number of bins used to section the DODs.

Fuel cell mode	Mode 1	Mode 2
Output [kWh]	0.5	0.25
Input [L/h]	≈ 0.285	≈ 0.114
η	0.4	0.5

Table 4.1: Fuel cell mode properties

$L(\xi_i)$ describes the number of discharge cycles a battery can withstand given a specific DOD and is described as

$$L(\xi) = a\xi + b$$

where a = -22467 and b = 2873 are empirically derived parameters fit to manufacturer cycle life data [35]. This model only accounts for cycle aging of the battery and disregards other reasons that might cause degradation in the battery.

4.1.4.1 Fuel cell model

The fuel cell model was assumed to have a constant output described in table 4.1. The outputs and inputs were considered to be constant within each timestep since the minimal temporal resolution of the simulation was 1 hour and nonlinear behaviors were assumed small and were neglected at that timescale. The input in liters were calculated from the output demand and efficiency by

$$\gamma \frac{p}{\eta} \frac{1}{LHV}$$

Where p is the output of the fuel cell described in table 4.1, η is the efficiency described in the same table, LHV is the lower heating value of the fuel and γ is a constant that transforms kW to MJ.

4.1.4.2 Solar panel

The solar panel model was implemented using a Python library called PVlib [2] which provide models for photovoltaic systems. A Sandia Array Performance Model (SAPM) was used to model the output. Panel output was scaled proportionally to the total installed surface area relative to the reference module area.

4.1.4.3 Wind turbine

The wind turbine is modeled using a piecewise power curve with a cut in and a cut off speed. Below the cut of speed the rotor was assumed to not generate any power and above the cut of speed the turbine is shut of to prevent mechanical damage. Between these limits the output scales cubically representing the cubic relationship between wind speed and kinetic energy. The turbine output was described as follows

$$P_{wt} = \begin{cases} 0 & \text{if} \quad \phi > \phi_{off} \quad \textbf{or} \quad \phi < \phi_{in} \\ r \cdot \frac{\phi^3 - \phi_{in}^3}{\phi_r^3 - \phi_{in}^3} & \text{otherwise} \end{cases}$$

where r is the rated power output, ϕ is the wind speed, ϕ_{off} and ϕ_{in} is the cut off and cut in speed respectively and ϕ_r is the rated wind speed.

Residential building model

The residential building model was modeled as a storage components described below

$$Q(t) = Q(t-1) + Q_{in}(t-1) - Q_{out}(t-1) + Q_{amb}(t-1)$$

Where Q is the heat stored in the building. Q_{in} is the heat supplied by the heating system, Q_{out} is the heat removed by cooling and Q_{amb} is the heat gain from the ambient temperature. The passive heat loss to the environment is modeled as

$$Q_{amb}(t) = \frac{T_{amb}(t) - T_{in}(t)}{R_{th}} \cdot \Delta t$$

where R_{th} is the thermal resistance of the building, T_{amb} is the ambient temperature, T_{in} is the current indoor temperature, and Δt is the timestep duration.

Heat pump model

The heat pump is modeled as a dispatchable conversion component that transforms electricity to heat. The model used to calculate the output of the heatpump was

$$Q_{in} = COP \cdot P_{el}$$

where COP is the coefficient of performance and P_{el} is the electrical power consumed. The COP was modeled as a fraction of the Carnot limit, following the approach of [27] described as

$$COP = \eta \cdot COP_{carnot}$$

where $\eta = \frac{1}{3}$ and COP_{carnot} is given as

$$COP_{carnot} = \frac{T_{cond}}{T_{cond} - T_{evap}}.$$

T_{cond} and T_{evap} was chosen to be $25°$ and $-5°$ respectively.

Air conditioning model

The air conditioning model was also modeled as a dispatchable conversion components. The air conditioning model uses the same model as the heat pump with the difference that the COP was calculated as

$$COP = \eta(\frac{T_{cond}}{T_{cond} - T_{evap}} - 1)$$

where the -1 term reflects the air conditioning removing heat from a cold storage rather than adding heat to the heat storage.

Electric vehicle (EV) charging station model

The EV charging station was modeled as a fixed conversion component. Individual cars would stay at the charging station and charge for a given amount of time and then leave. The schedule of arrivals and departures were given by creating a Markov decision chain by drawing probabilities from a Markov decision table.

Misc elec model

Some components of the load on the microgrid was set as a base load that were always committed. These devices did not need scheduling and therefore was not included in the optimizing process. Instead it was comprised into one category called misc elec and represented a fixed load on the system.

4.2 Genetic algorithm process

In this thesis a binary GA and a real coded GA is developed for the microgrid UC/ED scheduling problem. This encoding structure was chosen to fit the problem structure where in the mountain hut scenario the commitments of devices were binary and in the farm scenario they were continuous. The genes in each encoding contain information about the status or the power output of a component at a given point in time. Figure 4.1 show the workflow of the GA process.

4.2.1 Encoding

The genes in the chromosomes were ordered into a matrix where the rows represented the components and the columns represented the timesteps illustrated in Figure 4.2. This encoding was chosen to match the temporal structure of the problem.

4.2.2 Population size

A population size of 500 chromosomes was used across all experiments considering the mountain hut scenarios, and each test was terminated after 500 generations. This was selected as a balance between solution time and solution quality. For the farm scenario the test instead was terminated after 1000 generations.

4.2.3 Selection method

Deterministic tournament selection was used where n individuals were picked from the current population with replacement and the best one was chosen to be part of the parent pool. In this thesis, the selection pressure was tuned mainly by varying the size of the tournament pool where a value of 7 participants were used in the mountain hut and 2 were used in the farm scenario.

4.2.4 Crossover scheme

The crossover strategy used for the mountain hut scenario was a 2-point block wise recombination of the chromosomes. For the farm the 2-point crossover was used and compared to SBX-crossover.

4.2.5 Mutation scheme

The mutation operator for binary encoded genes implements a binary switching schema where a percentage of the population p_{mut} is chosen and within the chromo-

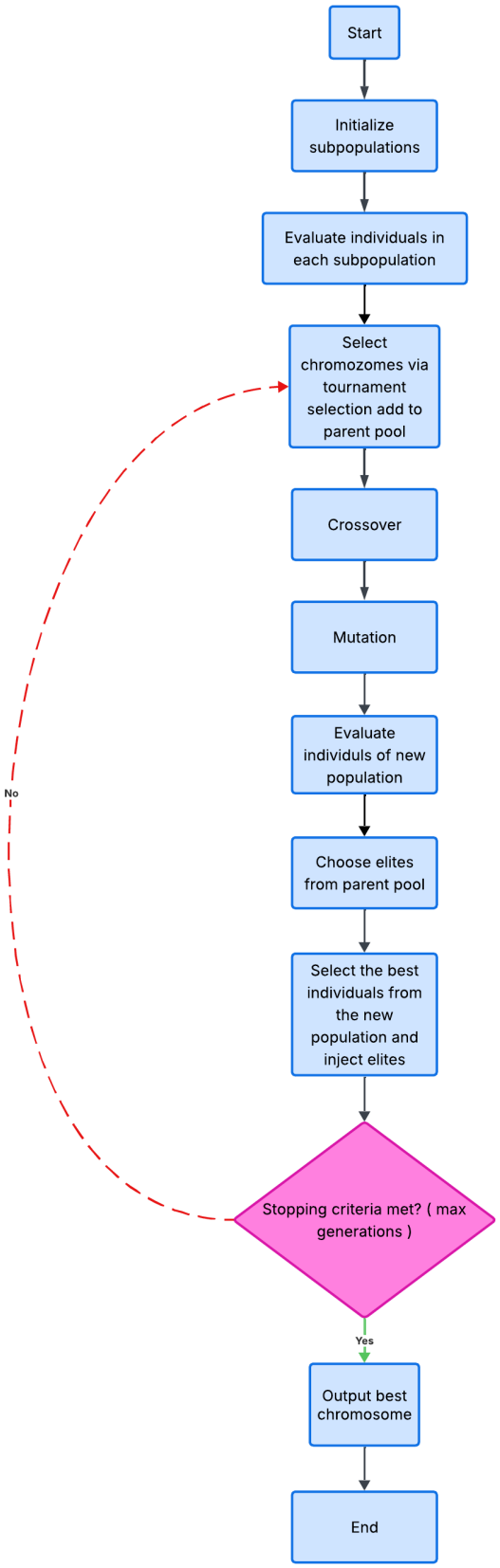


Figure 4.1: GA algorithm workflow

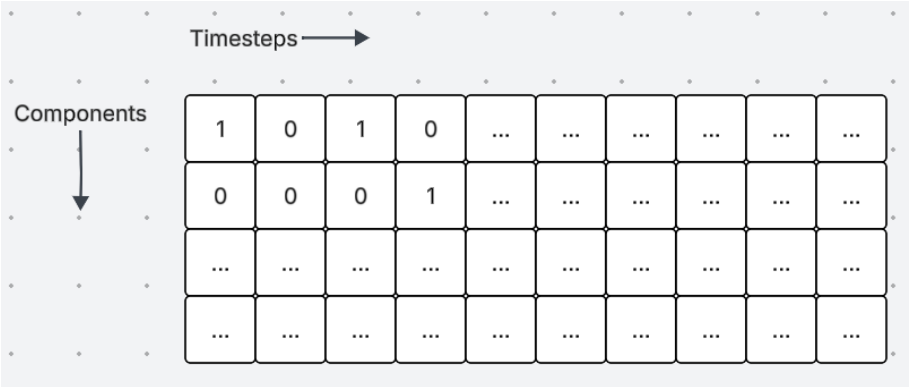


Figure 4.2: Chromosome representation

somes chosen for mutation a individual gene is chosen to be mutated with probability p_{ind}. The entire process can be described as

$$\text{if} \quad r < p_{mut} \quad x_n = \neg x_n$$

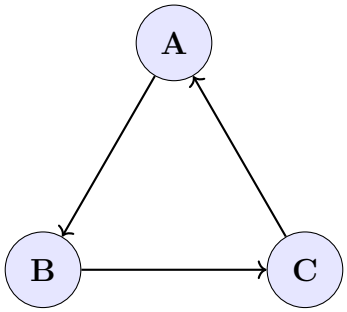
A population level mutation rate of 0.2 was chosen for the experiments and an individual mutation rate of $\frac{1}{L}$ was used where L is the length of the chromosome. An ablation study testing mutation values that were common in the literature can be viewed in the appendix.

4.2.6 Mutation used for real coded genes

The mutation operator for real coded genes used individual mutation rate of 0.01 and a population mutation rate of 0.2. A value was drawn from $\mathcal{N}(0, 0.05)$ and added to the gene selected for mutation. If the new values of the gene were outside of the allowed values they were clipped to be within 0 and 1. The real coded genes were initialized to a value between 0 and 1 so a mean perturbation would change the value of the gene by 5 %.

4.2.7 Evolution

In this thesis a migration ring setup was used as a structure for evolving the population. The migration ring is configured so that the entire population is divided up into 8 subpopulations which evolve separately. After 20 generations, 20 of the best individuals from a subpopulation migrate to another subpopulation. A circular migration is done so that in a three island setup the migration is performed as shown below.



4.2.8 Objective function

The objective function of the mountain hut was comprised of three objectives. Increaing uptime of the antennas, reducing size of the DODs of the battery and minimizing the fue usage. The goal of increaing uptime of the antennas were encoded as

$$\max \sum_{c} \sum_{t} c_{off} * p_{off}$$

where c are the components and p is their respective penalty per timestep. The goal of fuel consumption minimization was encoded as

$$\max \quad SOC_{f_T} - SOC_{f_0} = \max \quad \Delta SOC_f.$$

Where SOC_{f_T} is the fuel tank SOC at the last timestep and SOC_{f_0} is the fuel tank SOC at the first timestep.

4.2.8.1 Switching penalty

To reduce the excessive switching of the components a edge detector was used that detected falling and rising edges in the statuses of the components where a falling edge indicated a shut down of a components and a rising edge the starting of a component. I was encoded as shown here:

$$\max - \sum_{c} \sum_{t} |c_t - c_{t+1}|$$

4.2.8.2 Interval penalty

A penalty was added to the storage devices where a setpoint intervall was needed. The interval penalty was added to the storage devices to minimize the depth of discharge in an effort to minimize degradation costs. A minimum and a maximum setpoint value was added the reward/penalty for the storage device become active when the SOC went outside of this range.

$$\text{Interval penalty} = \begin{cases} \gamma \sum_{t} \left(\min(0, SOC_t - E_{\min})^3 + \min(0, E_{\max} - SOC_t)^3 \right) & \text{if } 0 \leq SOC_t \leq 1 \\ \psi\gamma \left(\sum_{t} \min(0, SOC_t - E_{\min})^3 + \min(0, E_{\max} - SOC_t)^3 \right) & \text{otherwise} \end{cases} \tag{4.1}$$

Where ψ was set to a large value. The case of a very large fixed penalty

$$\text{Interval penalty} = \begin{cases} \gamma \sum_{t} (\min(0, SOC_t - E_{\min})^3 + \min(0, E_{\max} - SOC_t)^3) & \text{if } 0 \leq SOC_t \leq 1 \\ -10^6 & \text{otherwise} \end{cases}$$

The gamma value was a tunable parameter and was set experimentally to balance the battery management with the other objectives.

4.2.8.3 Priority penalty

A penalty to prioritize the components of the mountain hut scenario was also used described as

$$\sum_{t}^{T} p \cdot (4G_{on}(t) \cdot (1 - 2G_{on}(t)))$$

where p was a gain factor. This penalty was added to enforce the the commitment prioritization order where the 2G antenna was committed before the 4G.
For the farm scenario interval penalties were used to penalize values of temperature that deviated from the interval as section 4.1.7.2.

4.2.8.4 Fitness function

The fitness function of the GA for the mountain hut penalized the up time of the components, fuel tank usage and battery SOC deviations

$$\max \sum_{c} \sum_{t} c_{off} * p_{off} + \Delta SOC_f + \text{Interval penalty} \tag{4.2}$$

and for the farm scenario

$$\max \sum_{t} grid_{\text{sell, t}} - grid_{\text{buy, t}} + \text{Interval penalty} \tag{4.3}$$

4.2.9 Warm start

A warm start was used for the Genetic algorithm for the mountain hut scenario consisting of a priority list. The warm start created a schedule and seeded the initial solution with one copy per island. The priority list algorithm created the schedule by computing the amount of energy available and checking if there was enough energy to keep the 2G antenna on. If it was possible for all timesteps the same was done for the 4G antenna.

4.2.10 Constraints

The constraints were handled using repair for hard constraints and penalties for soft constraints

The hard constraints on SOC limits of the fuel tank and battery were enforced via a repair step where a proposed solution was simulated and altered if a infeasible schedule was found. The effect of the proposed schedule on the storage components were evaluated and if a chromosome had settings that made the storage SOC drop below 0 % or go above 100 % then the components that were generating the deficit/excess were turned off. Other constraints such as the temperature constraints were handled by the termination criteria, where the solver did not terminate if the best solution violated the constraints.

4.2.11 Termination criteria

The termination criteria that was used for this thesis was a maximum number of generations set to 500 for the mountain hut scenarios and 1000 generations for the farm. The reason for choosing different termination criteria came from a converge study in A.5.

4.2.12 Hyperparameter tuning

The hyperparameter was tuned experimentally on the representative scenario with a simulation period of 336 hours and a time step of 4 hours. Because no specific setting of hyperparameter are optimal for every problem [34] the hyperparameter settings was guided by convergence speed, a tradeoff between exploration and exploitation as well as the ability to find solutions that were deemed desirable.

4.3 Mixed-Integer Linear Programming Formulation

The objective of the optimizer was to determine an optimal schedule of controllable components over a defined time space. The optimizer encodes physical operating constraints such as battery state of charge limits and energy balance requirements.

4.3.1 Sets, Indices, and Parameters

The following sets were defined:

- $T = \{0, 1, \ldots, |L| - 1\}$
- $\mathcal{C}_{\text{disp}}$
- $\mathcal{C}_{\text{fixed}}$
- $\mathcal{C}_{\text{on/off}}$
- $\mathcal{S}_{\text{bi}}$
- $\mathcal{S}_{\text{uni}}$
- $\mathcal{K}$

Where T is the set of discrete time steps, C_{disp} are the components that had a controllable continuous output. C_{fixed} are the components with fixed output values such as fixed loads in the system. $C_{on/off}$ are the components with controllable binary statuses like the antennas. S_{bi} and S_{uni} are the bi and uni-directional storage components and K is the set of energy carriers.

For each timestep t and component c time varying parameters were defined.

- $\text{out}_c(t)$
- $\text{in}_c(t)$
- $r_c(t)$
- $\sigma_c(t)$
- $\phi_c(t)$
- $w(t)$

Where $\text{out}_c(t)$ is the conversion efficiency output ratio of component c, $\text{in}_c(t)$ is the input consumption ratio, $r_c(t)$ is the rated power in kW, $\sigma_c(t)$ is the self discharge rate of storage components c, $\phi_c(t)$ is the extra SOC contribution from ambient conditions added to building models and $w(t)$ is the time-step weight that combines duration and a optional time-preference decay value.
The exponential decay weight was defined as:

$$w(t) = e^{-\alpha t} \cdot d(t) \tag{4.4}$$

where α is the decay rate and d(t) is the duration of time step at t. When no temporal discounting is desired, $\alpha = 0$ and the weights reduce to the pure time-step durations.

4.3.2 Decision Variables

The model contains three classes of decision variables:

Continuous variables For each dispatchable component and time step, the variable $f_c(t) \geq 0$ represented the continuous output level as a fraction of the rated capacity. For bidirectional storage, separate variables $f_{s,\text{in}}(t)$ and $f_{s,\text{out}}(t)$ represented charging and discharging power respectively.

Binary variables For each component and time step, the binary variable $y_c(t) \in \{0, 1\}$ indicates whether a component is on or off. For bidirectional storage, binaries $y_{s,\text{in}}(t)$ and $y_{s,\text{out}}(t)$ control weather the storage component is charging or discharging.

State of charge variables For each storage component, a continuous variable $\text{SOC}_s(t) \in [0, 1]$ represents the percentage stored energy relative to total capacity. A terminal variable $\text{SOC}_s^{\text{final}}$ captures the state at the end of the horizon which was used in the rolling horizon optimization mode to transfer the final SOC of the storage to the new optimizing window.

4.3.3 Constraints

Operating Range Constraints For each dispatchable component, power flow is bounded by the component's minimum and maximum operating range, enforced only when the unit is active:

$$P_{\text{min},c}\, y_c(t) \;\leq\; f_c(t) \;\leq\; P_{\text{max},c}\, y_c(t) \tag{4.5}$$

When $y_c(t) = 0$, both bounds force $f_c(t) = 0$, ensuring no output comes from a unit that is not active. The same structure applies to each direction of the bidirectional storage:

$$P_{\text{min},s}\, y_{s,\text{in}}(t) \;\leq\; f_{s,\text{in}}(t) \;\leq\; P_{\text{max},s}\, y_{s,\text{in}}(t) \tag{4.6}$$

$$P_{\text{min},s}\, y_{s,\text{out}}(t) \;\leq\; f_{s,\text{out}}(t) \;\leq\; P_{\text{max},s}\, y_{s,\text{out}}(t) \tag{4.7}$$

For unidirectional storage components, no binary variable is required and bounds are applied as simple box constraints.

State of Charge Dynamics The SOC of each bidirectional storage component evolves according the energy balance of the battery. For $t > 0$:

$$\mathrm{SOC}_s(t) = \mathrm{SOC}_s(t-1)\Big(1-\sigma_s\Big)+f_{s,\mathrm{in}}(t-1)\,\frac{r_s(t-1)}{E_s}\,\eta_s^{\mathrm{ch}}-f_{s,\mathrm{out}}(t-1)\,\frac{r_s(t-1)}{E_s\,\eta_s^{\mathrm{dis}}}+\phi_s(t-1),$$

where E_s is the total energy capacity, and η_s^{ch}, η_s^{dis} are the charging and discharging efficiencies. The initial condition of the SOC is based on a given value in the configuration for the first optimization window w and given by the final SOC of last optimizing window otherwise:

$$\mathrm{SOC}_{s,w,0} = \begin{cases} \mathrm{SOC}_{s,w-1,\mathrm{final\ t}_{w-1}} & \text{for} \quad w > 1 \\ \text{User set} & \text{for} \quad w = 1 \end{cases} \tag{4.8}$$

SOC bounds are enforced at every time step:

$$\mathrm{SOC}_{\mathrm{min},s} \;\le\; \mathrm{SOC}_s(t) \;\le\; \mathrm{SOC}_{\mathrm{max},s}, \tag{4.9}$$

with $\mathrm{SOC}_{\mathrm{min}} = 0.15$ and $\mathrm{SOC}_{\mathrm{max}} = 1.0$ for the battery. A terminal SOC constraint prevents end-of-horizon depletion:

$$\mathrm{SOC}_{\mathrm{min},s} \;\le\; \mathrm{SOC}_s^{\mathrm{final}} \;\le\; \mathrm{SOC}_{\mathrm{max},s} \tag{4.10}$$

Simultaneous charging and discharging is prevented by the mutual exclusivity constraint:

$$y_{s,\mathrm{in}}(t) + y_{s,\mathrm{out}}(t) \;\le\; 1 \tag{4.11}$$

For unidirectional (charge-only) storage, the dynamics simplify to:

$$\mathrm{SOC}_s(t) = \mathrm{SOC}_s(t-1)\Big(1-\sigma_s(t-1)\Big) + f_s(t-1)\,\frac{r_s(t-1)}{E_s}\,\eta_s + \phi_s(t-1),$$

Switching State Constraints For on/off components, a binary switching variable $s_c(t)$ captures state transitions in either direction:

$$s_c(t) \;\ge\; y_c(t) - y_c(t-1) \tag{4.12}$$

$$s_c(t) \;\ge\; y_c(t-1) - y_c(t) \tag{4.13}$$

By penalizing $s_c(t)$ in the objective, unnecessary switching is discouraged, representing start-up and shut-down costs or traits that are considered undesirable in the antenna activation pattern.

Mutual Exclusivity of Operating Modes Certain components, such as the fuel cell had mutually exclusive operating modes representing different efficiencies and outputs where at most one mode could be active at any given time. This was enforced by the constraint:

$$\sum_{c \in \mathcal{M}} y_c(t) \;\leq\; 1 \tag{4.14}$$

where $\mathcal{M}$ denotes the set of mutually exclusive mode components.

Load Ordering Constraints Operational priority logic requires that a lower-priority load can only be active if all higher-priority loads are already activated. For example:

$$y_{2\mathrm{G}}(t) \;\leq\; y_{\mathrm{misc}}(t) \tag{4.15}$$

$$y_{4\mathrm{G}}(t) \;\leq\; y_{2\mathrm{G}}(t) \tag{4.16}$$

Energy Balance Constraints The core requirement for a schedule to be feasible is that supply equals demand for every energy carrier at every time step.

4.3.4 Objective Function

The objective function represents the goal of the optimization. The primary objective of the mountain hut scenario is to maximize a weighted combination of antennas while penalizing fuel consumption and excessive switching:

$$\begin{aligned} \min \quad & -\lambda_1 \sum_{t \in \mathcal{T}} w(t)\, y_{\mathrm{misc}}(t) - \lambda_2 \sum_{t \in \mathcal{T}} w(t)\, y_{2\mathrm{G}}(t) \\ & -\lambda_3 \sum_{t \in \mathcal{T}} w(t)\, y_{4\mathrm{G}}(t) + \lambda_5 \sum_{t \in \mathcal{T}} w(t)\, f_{\mathrm{fuel}}(t) \\ & +\lambda_6 \sum_{c \in \mathcal{C}_{\mathrm{on/off}}} \sum_{t \in \mathcal{T}} w(t)\, s_c(t) \end{aligned} \tag{4.17}$$

For the farm scenario, the objective function goal was to minimize the cost of imported energy

$$\min \quad \lambda_1 \sum_{t \in \mathcal{T}} w(t)\, grid_{\mathrm{buy}}(t) - \lambda_1 \sum_{t \in \mathcal{T}} w(t)\, grid_{\mathrm{sell}}(t)$$

The weighting parameters λ_1–λ_6 are configurable by a user, enabling the user to trade off competing priorities. An SOC penalty term is also added: either penalizing usage of battery or distance from the 50% midpoint:

$$+\lambda_4 \sum_{t \in \mathcal{T}} w(t)\, \delta_{\mathrm{SOC}}(t), \qquad \delta_{\mathrm{SOC}}(t) \geq \left| \mathrm{SOC}_{\mathrm{bat}}(t) - 0.5 \right| \tag{4.18}$$

4.4 Evaluation metrics

The optimization with the GA and the MILP was run with the implementation described above. In order to compare optimization performance with each other a general evaluation metric was chosen called the ground truth metric. This metric was chosen to represent how well the optimization result fulfilled the desired outcome. For the mountain hut scenario it was chosen to account for the quality of service, energy cost and battery degradation. For the farm scenario the same metrics were used but fuel cost were exchanged for the grid interaction cost and the quality of service was modeled in another way described below.

Mountain hut quality of service The main objective of the mountain hut scenario was to provide coverage of telecommunications to its surroundings. Therefore the quality of service of the mountain hut was chosen to be a sum of the up time of the 2G and 4G antenna. In addition to the summation, time of day dependent weights were also included to account for that antenna activation can be more valuable during the day than during the night.

$$\text{QOS} = \gamma(h)(\sum_t \mu_1 2G_t + \sum_t \mu_2 4G_t)$$

The time of day weights were chosen as follows

$$\gamma(h) = \begin{cases} 1 & \text{if } 00.00 \leq h \leq 05.00 \\ 1.2 & \text{otherwise} \end{cases} \tag{4.19}$$

where h is the time of day. Individual weights were also added to the summation of the components where μ_1 and μ_2 represent these weights for the 2G and 4G antenna. These were added to emphasize that the 2G antenna were deemed to give a more valuable connection as it provided connection to emergency services while requiring less energy to run compared to the 4G. These represent the importance of the individual component where the ratio of the penalties of the two components had to be larger than the ratio of their energy expenditure.

$$\frac{\text{2G weight}}{\text{4G weight}} > 2 \tag{4.20}$$

This is because the energy ratio between the two components were given as

$$\frac{2Gantenna_{kWh}}{4Gantenna_{kWh}} = \frac{0.240}{0.122} \approx 1.96.$$

For the optimizer to value one time step of activation of the 2G antenna over two time steps of activation for the 4G antenna the ratio in Equation 4.20 must hold.

Farm quality of service The main goal of the farm scenario is the ability to keep individual buildings within certain temperature limits. The quality of service therefore becomes a quantification of how well this is fulfilled. In certain scenarios the buildings that are being controlled can contain the harvest from a given season.

If the temperature goes outside of the defined ranges the entire harvest will spoil and have a huge negative economic impact. Other buildings may just be residential buildings where the cost of small deviations outside of the constraints isn't as costly. To include this into the quality of service a large negative value is added to the ground truth metric if values are outside of the allowed temperature ranges for cold storage buildings. Smaller cumulative penalties are added for warm storage buildings.

Fuel cost Fuel cost was calculated based on the difference between the initial level and the final level of fuel in the tank times the cost of fuel given as

$$\text{Fuel cost} = f_{cost} * (ft_T - ft_0).$$

The cost of fuel per liter was set to a fixed value of 0.436 euro [15]. The battery degradation calculation is described in Section 4.1.4

Grid cost

The cost of interaction with the grid was modeled as

$$G_{cost} = \text{profit from sold energy} - \text{cost from bought energy}$$

Ground truth metric (GTM) The ground truth metric of the mountain hut consisted of 3 terms and 3 weights that combines quality of service, fuel cost and battery degradation

$$GTM = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3$$

x_1 represents the quality of service and α_1 was the weight given to the quality of service. The value of this weight is difficult to determine since it is not clearly defined how the quality of service exactly relates to the cost of the system. The function of the mountain hut is to provide good coverage of telecommunication, which is why the up time of the components were needed to be valued more highly than fuel minimization in the objective function. This was expressed as

$$\sigma = \frac{\text{value}_{1kWh,4G}}{\text{value}_{1kWh,fuel}} >> 1.$$

Simplifying the expression leads to

$$\frac{\text{value}_{1h,4G}}{0.122} = \sigma \cdot \text{value}_{L,fuel} \cdot (\frac{LHV}{3.6} \cdot \eta)^{-1}$$

$$\text{value}_{1h,4G} = \sigma \frac{0.122 \cdot 3.6}{\eta LHV} \cdot \text{value}_{L,fuel}$$

where 0.122 is the energy demand of the 4G antenna per hour, 3.6 is a conversion factor between [MJ] and [kWh], η is the lowest efficiency of the fuel cell and LHV is the lower heating value. This simplifies to

$$\text{value}_{1h,4G} = 15 \cdot 0.57 \cdot 0.122 \cdot \text{value}_{L,fuel}$$

Where 0.57 is rounded up to 0.6. The $\text{value}_{1h,4G}$ becomes the α_1 weight.

$$\alpha_1 = 15 \cdot 0.6 \cdot 0.122 \cdot 0.436.$$

x_2 represents the fuel used and α_2 the cost of the fuel.
α_3 is the weight assigned to the battery degradation cost, set to the approximate cost of a new battery. This is calculated as the cost per kWh of battery capacity multiplied by the battery size used in the tested version of a scenario. When multiplied by x_3 which defines the relative degradation of the battery over the scheduling horizon gives an estimated value of the degradation cost produced by the optimized schedule.
The resulting ground truth metric therefore becomes

$$GTM = 15 \cdot 0.6 \cdot 0.122 \cdot 0.436 \cdot x_1 - 0.436 \cdot x_2 - 324 \cdot \text{battery size} \cdot x_3$$

For the farm scenario the resulting ground truth metric instead becomes

$$GTM = -324 \cdot \text{battery size} \cdot x_3 + G_{cost} - \gamma QOS_{farm}$$

where γ is a scalar indicating if there were a deviation outside of the allowed temperature limits.

4.5 Experimental setups

To test the optimization algorithms, 4 versions of the mountain hut scenario were generated and three versions of the farm scenario with varying configurations. Each version were then run 10 times for each hyperparameter setting varying only the random seed in the optimizer.

4.5.1 Mountain hut

The mountain hut represents an off grid unit that provides a 2G and 4G connectivity to its proximity. The usefulness of the mountain hut comes from its flexibility where it can be placed and provide telecommunication support where it is difficult for regular service infrastructure to be placed. The mountain hut consists of a battery, solar panels and wind turbines, a fuel cell capable of running in two efficiency modes and a fuel tank. Several other miscellaneous electrical components are also included in the setup modeled as a fixed load. The fuel tank is refueled twice a year in spring time. The mountain hut uses two energy carriers, fuel and electricity. The optimizers were tested on 4 versions of the mountain hut, listed in Table 4.2.

Complexity In the mountain hut scenario the word complexity is used to describe two scenarios:

- Complexity 0 - Indicating that the simulation is run with energy enough to commit all components.
- Complexity 2 - Indicating that the simulation is not run with enough energy to commit all components and therefore needs to make decisions about which components to commit and when.

Mountain hut version	0	1	2	3
Location	Sarek, Sweden	Sarek, Sweden	Sarek, Sweden	Sarek, Sweden
Season	Summer	Summer	Winter	Spring
Battery size [kWh]	40	5	25	7
Wind turbine	1	1	1	0
Solar panel area [m^2]	3	1.9	3.6	1.1

Table 4.2: Experimental mountain hut versions

Farm version	0	1	2
Location	Gotland, Sweden	Gotland, Sweden	Gotland, Sweden
Season	March	January	June
Battery size [kWh]	16	35	46
Wind turbine	0	0	2
Solar panel area [m^2]	30	50	10

Table 4.3: Experimental farm versions

4.5.2 Farm

The farm scenario is a more flexible scenario compared to the mountain hut in that it can be used to represent different objectives. One such objective could be precise heating or cooling of a building or the scheduling of demand of EV charging to minimize the charging cost. The farm scenario also has a grid connection, allowing it to buy and sell energy in interactions with the energy market. The farm uses two energy carriers, electricity and heat. Like the mountain hut scenario it has a battery, and a solar panel. It also uses components for cooling and heating and has an EV charging station drawing a fixed load at random intervals. The optimizer were tested on 3 versions of the farm, listed in Table 4.3

4.6 Software used

The project was carried out entirely in Python. Python specific software was used for optimization algoritm implementation and for calculating the output of the solar panels.

4.6.1 Distributed Evolutionary Algoritms in Python (DEAP)

DEAP is a flexible framework implemented in python that makes it easy to create custom evolutionarily algorithms. The frameworks supports metaheuristic computation techniques such as genetic algorithms, swarm based intelligence algorithms, genetic programming, evolution strategies and differential evolution. In this thesis it was used for the creation of a genetic algorithm and the custom implementations of genetic algorithm operators such as selection, mutation and crossover.

4.6.2 Pyomo

Pyomo is a python library that enables for the creation of linear optimization programs.

4.6.3 Pvlib

Pvlib is a framework where you can create custom setups of solar panels and calculate the output from them based on panel positioning, position of the sun and the reported values of solar irradiance.

4.6.4 OpenMeteo

The weater data used in the tested scenarios was downloaded from OpenMeteo which is an open source weather service providing historical weather information for a variety of locations as well as a historical weather forecasts. The historical forecast has a forecast lag of up to 16 days.

4.6.5 Ensoe-e

Entsoe-e was used to provide information about historical grid spotprices. This service was used in the modeling of interactions with the energy market. No uncertainty was modeled in the price of energy. Sell and buy prices were modeled from the same spotprice but with an arbitrary tax added on the purchase price.

5

Results

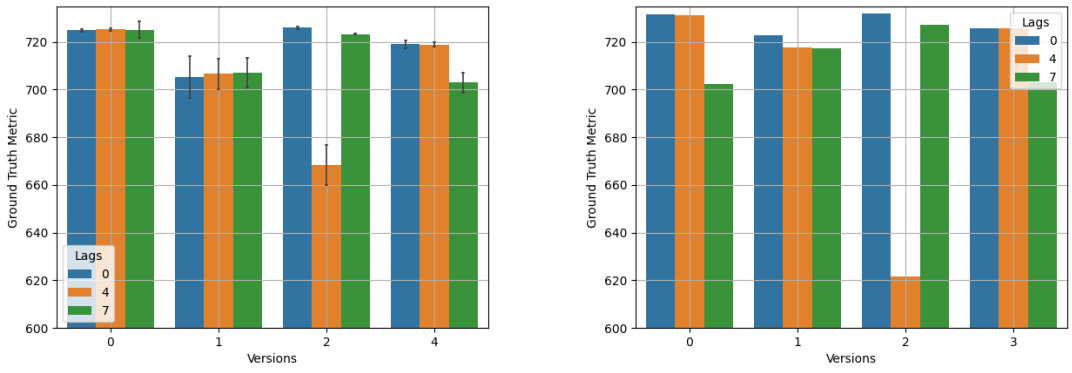
5.1 Optimality

The following section evaluates the optimizers on performance based on the GTM.

5.1.1 Mountain hut

Below follows the results from the mountain hut tests.

To investigate how the optimizers performed on different versions of the mountain hut under ideal conditions with full initial fuel tank and with varying levels of uncertainty in the forecast, tests were run to analyze fitness and optimizer behavior across different configurations of the mountain hut.



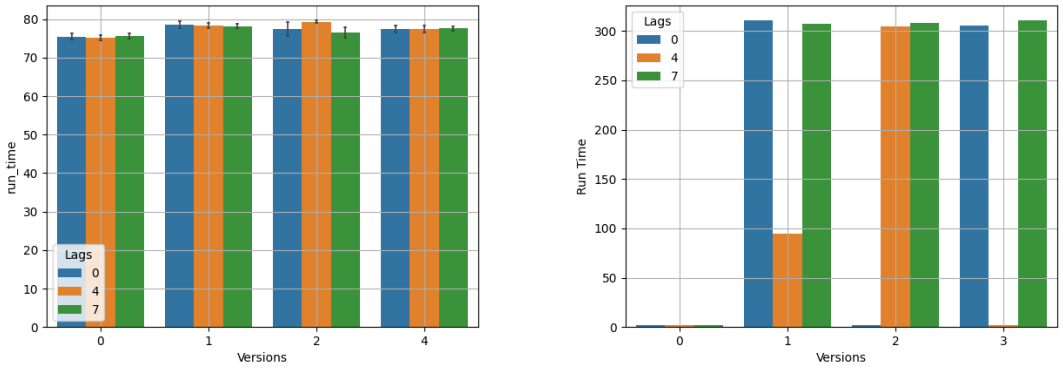
(a) GA performance on Ground truth metrics

(b) MILP performance on ground truth metric.

Figure 5.1: Ground truth metrics visualization across mountain hut versions and forecast lag. Complexity 0.

Mountain hut version	0	1	2	3
GA ground truth metric score	725	708	726	719
MILP ground truth metric score	731	722	732	724
Gap %	0.8	1.9	0.8	0.7

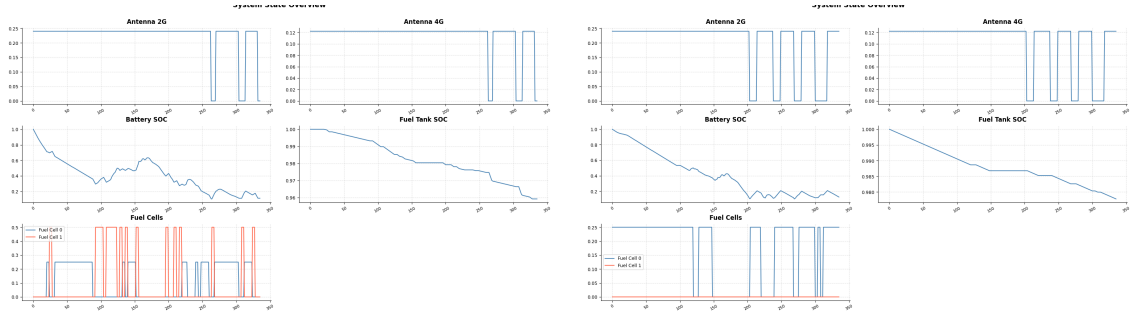
Table 5.1: Ground truth metric value comparison between GA and MILP across four versions of the mountain hut. No forecast error.



(a) GA run time performance

(b) MILP run time performance

Figure 5.2: Computational performance compared across mountain hut versions, forecast lags and optimizers.



(a)

(b)

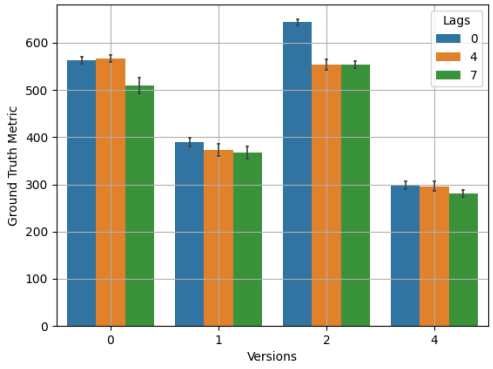
Figure 5.3: Visualization of GA (a) and MILP (b) schedule outcome for mountain hut version 2 and forecast lag 4, complexity 0. The plots show, starting from the top row going from left to right: 2G antenna activation, 4G antenna activation, battery SOC, fuel tank SOC and fuel cell activation where the red and blue color indicate different efficiency and output modes of the fuel cell.

Table 5.1 show the specific values that the optimizers achieved with zero days forecast lag and the optimality gap between the GA and MILP performance, plotted in Figures 5.1a and 5.1b. The gap between the GA and the optimal solution lie below 2 % for all versions and below 1% for most versions with a mean gap across versions of 1.05 %.

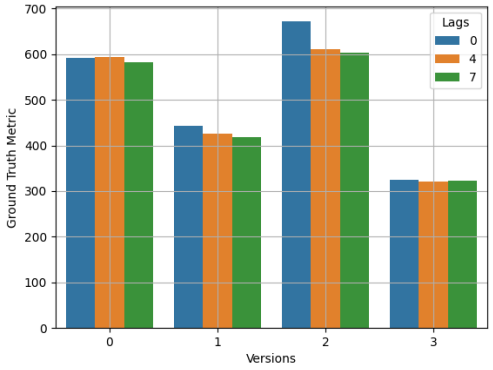
Figure 5.1a and 5.1b show that in a situation where there is no restriction on fuel usage, the ground truth metric performance is not affected very much with increasing forecast uncertainty. The exceptions are mountain hut version zero and version two where in version two the solution deteriorated both for the GA and the MILP and in version zero the solution deteriorated for the MILP only, visualized in Figure 5.3.

5.1.2 Optimizer behavior for increasing complexity

Having established optimizer performance under ideal conditions for increasing forecast uncertainty, tests were run with a reduced initial fuel level to compare optimizer behavior when energy availability was limited, creating a more complex optimization problem.

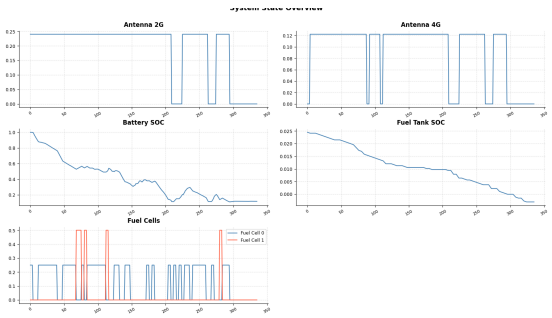


(a) GA ground truth metric performance.

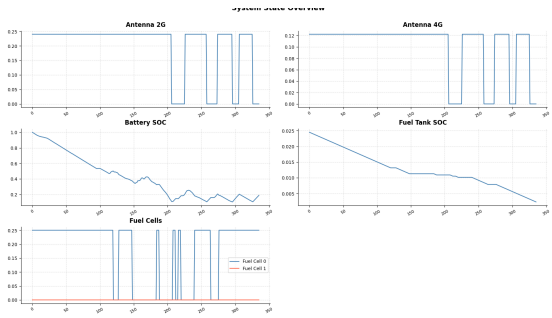


(b) MILP ground truth metric performance.

Figure 5.4: Ground truth metric performance with complexity 2.



(a) Visualization of GA schedule for mountain hut version 2 forecast lag of 4 days



(b) Visualization of MILP schedule for mountain hut version 2 forecast lag of 4 days.

Figure 5.5: Ground truth metrics performance with complexity 2

Mountain hut version	0	1	2	3
GA ground truth metric score (highest out of 10 runs)	559	389	641	298
MILP ground truth metric score	590	444	670	325
Gap %	5	12	4	8

Table 5.2: Optimality gap comparison between GA and MILP across four versions of the mountain hut. Zero days forecast lag.

Mountain hut version	0	1	2	3
GA ground truth metric score (highest out of 10 runs)	563	374	552	296
MILP ground truth metric score	590	425	608	320
Gap %	4.5	12	9	7.5

Table 5.3: Optimality gap comparison between GA and MILP across four versions of the mountain hut. Four days forecast

Mountain hut version	0	1	2	3
GA ground truth metric score (highest out of 10 runs)	506	368	552	281
MILP ground truth metric score	580	418	603	323
Gap %	13	12	8.4	13

Table 5.4: Optimality gap comparison between GA and MILP across four versions of the mountain hut. Seven days forecast lag.

Figures 5.4 show that the MILP and the GA perform similarly across lags compared to 5.1. This indicates that with limited energy, the sensitivity of the solution quality to weather uncertainty is lower. The optimizers both perform worse for increasing forecast lags across every version of the mountain hut with the exception of version 0 where both optimizers performed better on the four day forecast lag. This is counterintuitive for the MILP because it should not be able to produce a better schedule for a more uncertain forecast. Tables 5.2, 5.3 and 5.4 show that the mean optimality gap increased between the MILP and the GA compared to Table 5.1. The gap also increased for increasing forecast uncertainties. It also shows that the MILP consistently achieve a higher ground truth metric scores compared to the GA.

5.1.3 GA behavior for seeding initial population

The degradation of GA performance compared to MILP on the more complex problem is due to premature convergence of the GA. This can possibly be due to a large solution space making it more difficult for the GA to find and explore the correct area of the fitness landscape before population diversity shrinks. A way of counteracting problems with premature convergence is to seed the initial population with a guessed good solution guiding the GA to easier and quicker find an optimized schedule. A priority list seeding was tested with different initializations and evaluated on the impact on ground truth metric scores. The increased performance is also visualized with convergence curves.

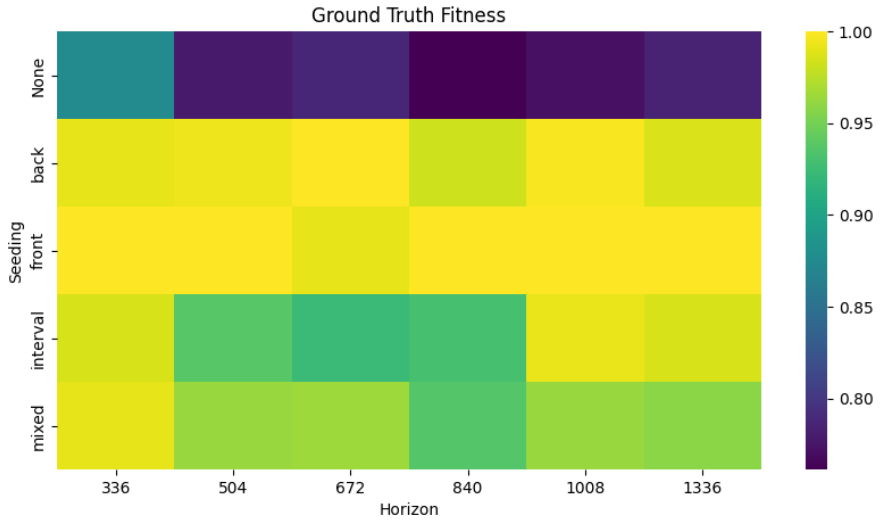


Figure 5.6: On the y axis: Seeding variants. On the x axis, simulation horizons. Each column is normalized by dividing each value by the maximum value achieved in that simulation horizon.

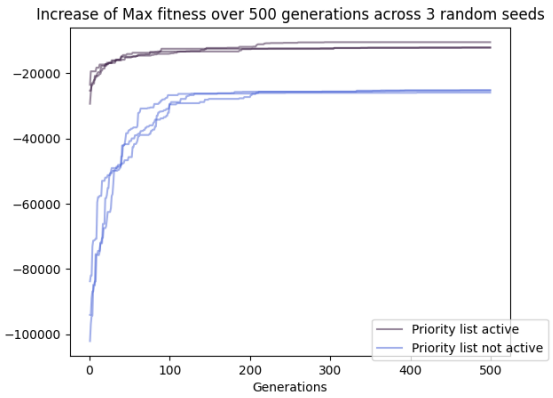


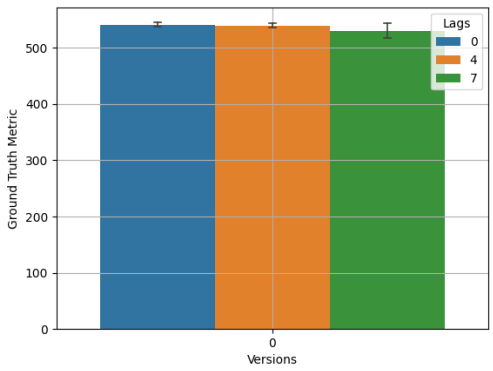
Figure 5.7: GA convergence curves. Mountain hut version 2, no forecast error, limited initial fuel.

Figure 5.6 shows the ground truth metric of the GA for 4 different priority list seeding variants and one where no seeding is used. The back seeding creates a schedule where the highest valued component is added to the last timestep and then to the second last timestep and so on. The front seeding functions similarly but adds commitments starting at the first timestep. The interval seeding adds components in intervals of 8 hours starting from the first timestep and the mixed adds both 2G and 4G at the same timestep starting from the first timestep. Figure 5.6 shows that using any priority list seeding increases the ground truth metric score in every tested simulation horizon where a front seeding performs the best, closely followed by the back seeding method. It also shows that no large deterioration of GTM occurs for the case with no seeding. This is somewhat surprising as a large time horizon grow the solution space possibly making it harder for a randomly initialized

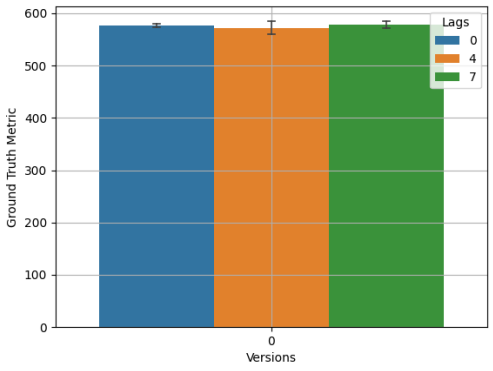
GA to find the near optimal solution. Figure 5.7 shows the convergence behavior for the GA with and without a priority list. Without the priority list the GA gets stuck in a local optima close to where the priority list seeded optimization begins the optimization process.

5.1.4 Comparing GA rolling horizon with varying time steps versus a fixed timestep.

To examine wether the GA performance improves using a shorter variable timestep duration tests were run comparing different setups. One setup uses a time step of 2 hours for the first 24 hours and a timestep of 8 hours for the remaining simulation period. The other setup uses a timestep of 8 hours throughout the simulation. Both tests uses rolling horizon optimization and is reoptimized after 24 hours using a planning horizon equal to the full time horizon.

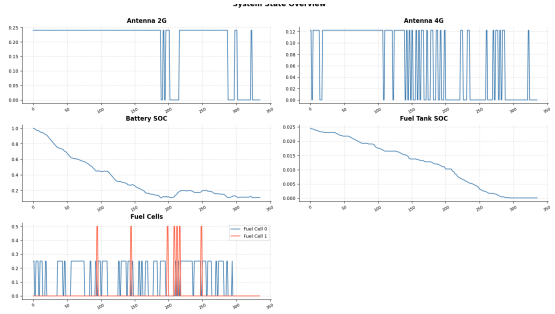


(a) GA ground truth metric performance on mountain hut 0 with a variable timestep of 2 hours for the first 24 hours and 8 for the remaining optimization. Complexity 2.

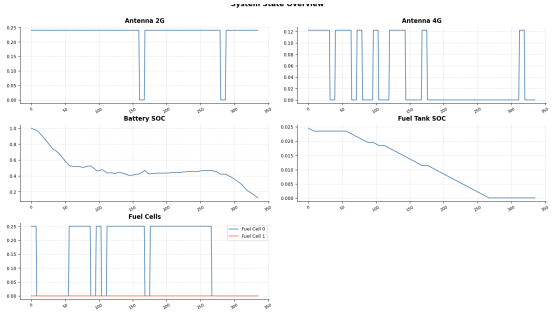


(b) GA ground truth metric performance on mountain hut 0 with a fixed timestep of 8. Complexity 2.

Figure 5.8: Comparison of ground truth metric comparing fixed vs varying time step



(a) Visualization of a well performing schedule solution achieved by the GA with variable timestep settings.



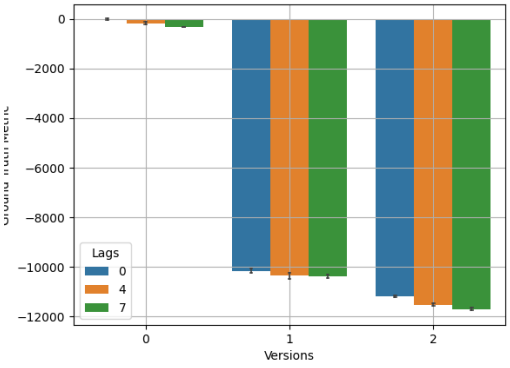
(b) Visualization of a well performing schedule solution achieved by the GA with fixed timestep settings.

Figure 5.9: Visualization of schedules comparing fixed vs varying timestep

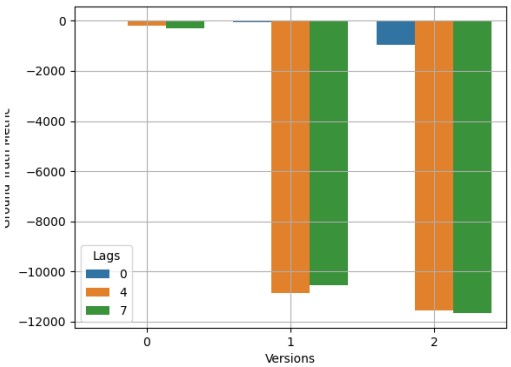
Figure 5.8a shows worse performance in ground truth metric compared to a timestep duration of 4 hours shown in Figure 5.4a and compared to a fixed timestep of 8 hours shown in Figure 5.8b. Notably the ground truth metric scores are higher across all forecast lags in Figure 5.8b compared to version zero in 5.4a. Figures 5.9a and 5.9b show the produced schedules of the two best optimization runs. A clear pattern of increased switching behavior is evident in Figure 5.9a likely due to the possibility of the optimizer to create switching behavior and not fully converging due to a longer chromosome. The battery SOC limits are also more robust in 5.9b likely due to a more optimized 4G scheduling, fuel cell usage and 2G scheduling.

5.1.5 Farm

To investigate how the optimizers performed on the farm scenario tests were run on several version of the farm with a varying number of buildings in need of temperature regulation.

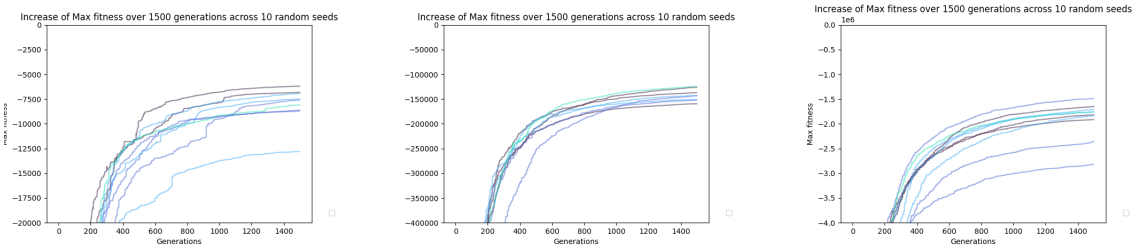


(a) GA ground truth metric performance for farm scenario version 0, 1 and 2



(b) MILP ground truth metric performance for farm scenario version 0, 1 and 2

Figure 5.10: Ground truth metric comparison



(a) GA convergence curve for farm version 0

(b) GA convergence curve for farm version 1

(c) GA convergence curve for farm version 2

Figure 5.11: Convergence curve comparison

Figures 5.10a and 5.10b show that the MILP performs better than the GA for zero day forecast lag while it performs similar or worse than the GA for a forecast error of 4 or 7 days. It also shows that performance deteriorates for increasing forecast error for both optimizers which is a result of that a significant part of the negative scoring come from the deviation of temperature outside of the allowed constraints. This can happen easily when the temperature constraints are narrow and the forecast errors are large. Figures 5.11 show that convergence is not achieved for optimizer runs on mountain hut version 0, 1 and 2. Together with Figure 5.10a this indicates that the GA needs to optimize for more generations to find acceptable solutions or use other improvements that increase solution quality.

The poor convergence and results of the GA given in Figure 5.10a could be a result of a non optimal penalty applied to the building temperature constraints. Therefore tests were run on scenario 1 with a quadratic set point penalty on building temperature instead of a cubic interval penalty, resulting in the results shown in Figure 5.12. It shows that the GA keeps the temperature constraints for all included buildings on a 24 hours simulation horizon which was not the case in Figure 5.10a, indicating that penalty tuning greatly affect the outcome of the optimization. For longer horizon the results start to deteriorate which indicates that the GA still struggles to keep temperatures constrained when chromosomes become longer.

5.1.6 Pareto front

To investigate how well the optimizers handles competing objectives a Pareto front was created illustrating the tradeoff between energy cost minimization and battery degradation by varying the penalty associated with battery degradation.

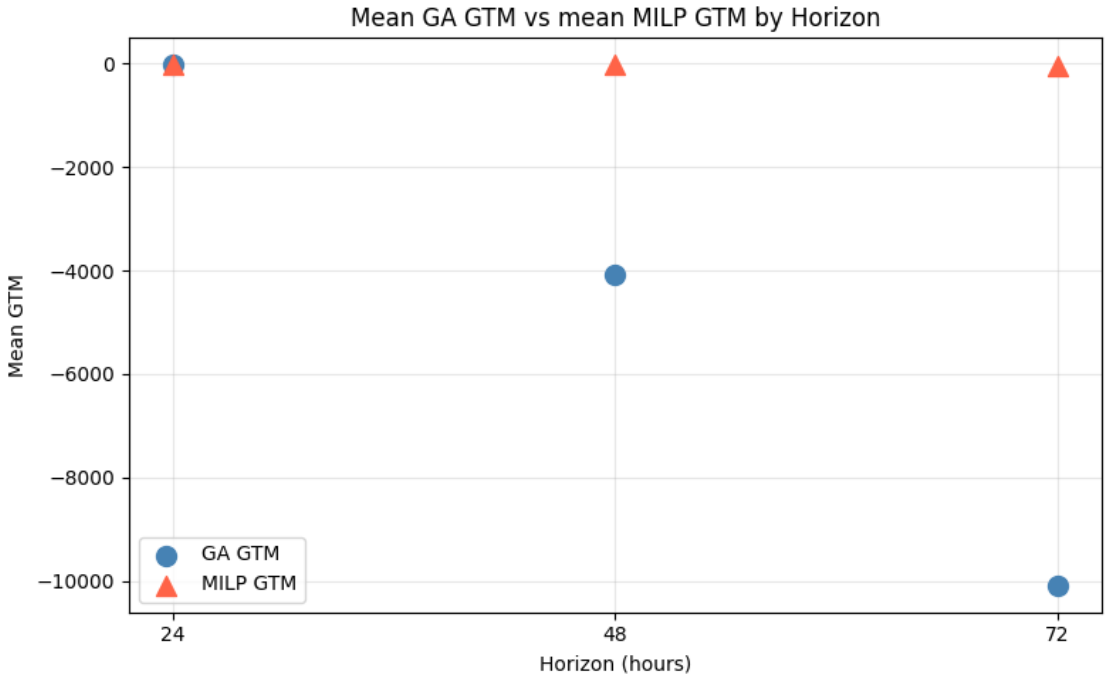


Figure 5.12: Ground truth metric for MILP and GA on simulation horizon 24, 48 and 72 hours with updated penalties for the GA. 1 day forecast error.

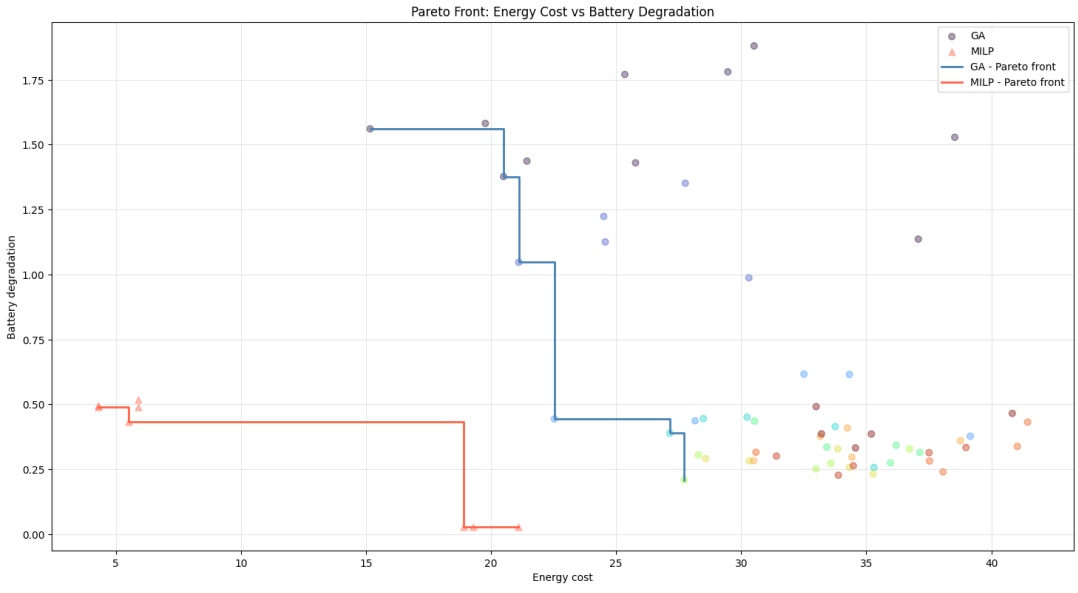


Figure 5.13: Pareto front illustrating the tradeoff between energy cost and battery degradation for GA and MILP. GA is run 5 times per penalty settings. Point colors represent different penalty settings for the battery degradation.

Figure 5.13 shows that increasing the cost of battery degradation induces a Pareto front for both optimizers where, when the battery degradation is valued highly, energy is bought at a higher price to reduce battery degradation. The MILP also

produces better results compared to the GA both in energy cost and battery degradation minimization.

5.2 Scalability

Below follows a investigation of the scalability of the MILP compared to the GA where the optimizers where run on longer simulation horizons and therefore using an increased number of variables.

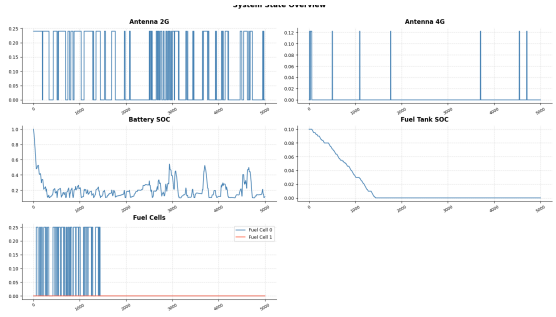
5.2.1 Optimizer behavior for longer time periods

Based on the finding that a lower resolution timestep was beneficial for computation time and ground truth metric performance for the GA on the mountain hut scenario, shown in the timestep ablation study in A.4, tests were run comparing the runtime and optimizer performance for a simulated time period 2000, 3000 and 5000 hours with a timestep of 10 hours.

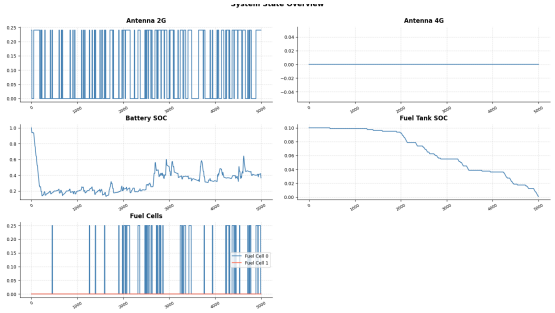
Horizon (h)	2000	3000	5000
GA ground truth metric score (best out of 10 runs)	1984	2405	3255
MILP ground truth metric score	2006	2413	3261
Gap %	1	0.3	0.1
Gap (absolute)	22	8	6
Run time GA (s)	291	428	706
Run time MILP (s)	300	300	300

Table 5.5: Optimality gap comparison between GA and MILP for mountain hut version 0, no forecast error, limited initial fuel and increasing length simulation horizons.

Table 5.5 show the optimality gap between MILP and GA for long time periods lie below or equal to 1 % showing that the GA achieves a solution close to the one produced by MILP for long simulation periods. It also shows that the optimality gap values between the GA and the MILP shrink as the simulation period increase indicating that the probem of premature convergence is less for long simulation horizons.



(a) Visualization of the schedule produced by the GA for 5000 simulated hours.



(b) Visual of the schedule produced by MILP for 5000 simulated hours.

Figure 5.14: Visualization of long simulation horizons

Figure 5.14a and 5.14b visualize the schedule created by the GA and the MILP. The schedules are different but produce similar ground truth metric scores.

5.3 Practical usability

The practical usability of the optimizer is of high importance as an optimizer that is difficult to use in practice will return worse results. Below follows a hyperparameter count and a hyperparameter sensitivity study of the GA on the farm scenario.

5.3.1 Hyperparameter count

One practical aspect of an optimizer is how many parameters are needed to be tuned in order for the optimization to work. Another important aspect is how sensitive the solution is to the settings of the hyperparameters. The number of hyperparameters in the GA can be divided up into sections described below:

5.3.1.1 Chromosome hyperparameters

- Population size
- Crossover rate
- Individual mutation rate
- Population mutation rate
- Mutation perturbation size
- Elitism count
- Tournament size
- Number of island
- Migration rate
- Number of individuals migrating

5.3.1.2 Method choices

- Crossover method

- Mutation method
- Selection method
- Warm start method
- Repair

5.3.1.3 Termination criteria choices

- Generations before termination

5.3.1.4 Penalties

- Constraint penalties
- Objective penalties

Between 14 and 15 hyperparameters, 2 constraint penalties and a minimum of 2 objective penalties require tuning/choosing across the two different scenarios for the GA.

For the MILP the objective weights require tuning which, for the scenarios considered is at least 3 and can increase depending on the problem size.

5.3.2 Hyperparameter sensitivity

To investigate the hyperparameter sensitivity and in extension the ease of use of the GA, tests were run on the farm using varying mutation and crossover levels.

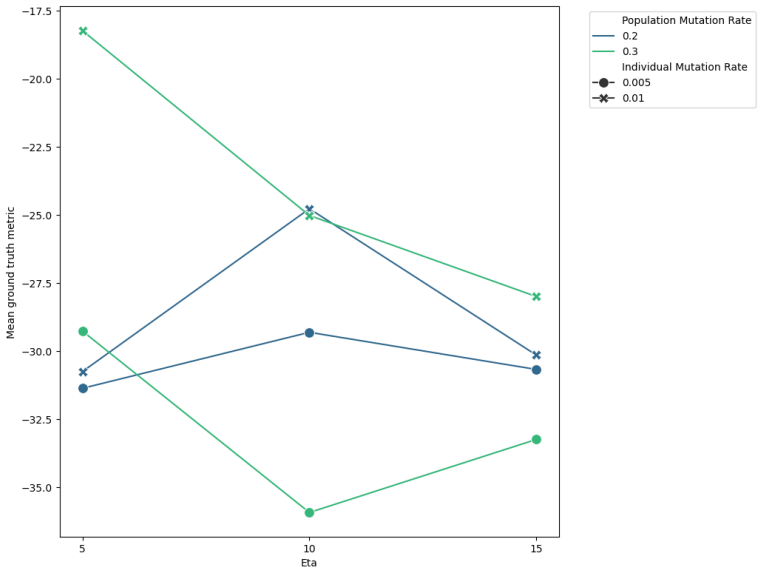


Figure 5.15: Population mutation vs individual mutation vs η parameter rate

Figure 5.15 shows that the performance varies significantly with varying hyperparameters with the best performance given by a large mutation rate and a low η

for the farm. The worst performance does not come from the opposite configurations but rather a setting with a lower individual mutation rate performing approximately twice as bad as the best configurations. Figure 5.15 further demonstrates that the performance of different population mutation rates depends on the value of η, suggesting the presence of interactions between hyperparameters. Consequently, a hyperparameter configuration that performs well under one set of conditions may perform poorly under another. This highlights the importance of carefully tuning the hyperparameters based on the characteristics of the addressed optimization problem.

6
Discussion

6.1 Forecast inaccuracy and penalty settings

Section 5.1.1 show that the GA produces better scores compared to the MILP for some cases of forecast errors on a one shot optimization. Figure 5.3b illustrates that this behavior occurred due to the objective weighting factors in the MILP setup resulting in a SOC trajectory with a smaller margin above the SOC cut off point compared to the GA. The SOC penalty for the MILP was determined by weighting parameters in the objective function where a penalty weight on absolute use of the battery was applied. This does not motivate the MILP to create a schedule that keeps the SOC limit high which resulted in the SOC dropping below the cut of point and forcing the shut down of the components. The same behavior is evident in the GA solution visible in Figure 5.3a where forced shut down of components also occur but to a lower degree. In the GA, SOC values that deviate outside of allowed limits are penalized quadratically which creates a strong increasing penalty as the GA approaches the cutoff. This discourages the GA from creating schedules close to the boundary, creating a more robust SOC trajectory. The increased robustness behavior of the GA for certain versions is explained by a difference in penalty setting between the GA and the MILP. This is supported by the results as the GA consistently have higher fuel costs and battery degradation costs across versions with zero complexity indicating that the penalty setting for the GA values a higher SOC margin in favor of fuel minimization or minimizing battery degradation. One way of increasing the robustness of the solution would be to use a set point penalty for the MILP. Another way is by using re optimization with a rolling horizon where much of the negative effects caused by the forecast errors are negated.

6.2 Simulation artifact

The counterintuitive result observed in Figure 5.4b likely stem from a simulation inaccuracy that is achieved by having a simulation step of 4 hours and a zero order hold update rule. The increased step size gives the MILP knowledge about the SOC values and the renewable generation as the mean over a 4 hour period which is not perfectly accurate. The zero order hold update also means that all of the energy generated in a timestep is assumed to be available at the initial point in that timestep, which might not be the case, creating a further inaccuracy of the true system state which can lead to the MILP creating a suboptimal solution where the optimization with increased forecast lag has a chance to achieve a better schedule.

6.3 Premature convergence

The deterioration in GA solution quality compared to the MILP for increasing problem complexity and forecast lags shown in Table 5.2, 5.3 and 5.4 can be attributed to the increase in solution complexity where the optimal solution may become more specific. This requires the GA to find a specific combination of on/ off switches that may be difficult to achieve through stochastic search. As the GA searches the solution space it is rewarded for activating components that contribute positively to the objective. This creates a bias that can trap the optimizer in a local optima, leading to premature convergence where escaping requires simultaneous changing of a specific subset of components. This makes it more unlikely for the GA to find the near optimal solution when faced with increasing solution complexity.

One way to mitigate the random switching of components and problem of premature convergence is through informed initialization. Using a priority list seeding gave improved results for the mountain hut case as shown in Figure 5.7 and 5.6, suggesting that guiding the GA towards a good solution estimate reduces the problem of premature convergence. However, seeding alone does not fully eliminate the problem as mutation and crossover can reintroduce switching. A switching penalty was introduced into the objective function in an effort to discourage switches as shown in A.2. However it did not yield any additional improvements to solution quality. One interesting result shown in Figure 5.6 is that when the solution space increases the solution created without a priority list does not deteriorate much compared to a shorter simulation horizon. This should reasonably be the case as a larger solution space makes it more probable that the GA will get stuck in a local optima earlier on.
Another possible solution to reduce the problem of premature convergence is to introduce swap mutation instead of bit flip mutation once the sub optimal solution have been found. Swap mutation would have a larger chance to create new feasible schedules since the activation of two components is swapped making it more likely that the same amount of energy is used for the mutated schedule.
Another case where the GA struggles is shown in Section 5.1.5 where the GA is unable to find a schedule that does not deviate outside of allowed temperature ranges even without forecast error. This is likely because of a combination of slow convergence shown in 5.11 and narrow constraints on temperature. One solution would be to have the GA continue optimizing until the constraints have been met although the additional computation time is hard to predict and may not be doable in practice.

6.4 Fitness space

The decreasing optimality gap shown in Table 5.5 may be attributed to the structure of the fitness space at longer time horizons. As simulation time increases the limiting factor is the available energy resulting in many activation patterns that produce similar ground truth metric scores. This makes the GA perform relatively better

compared to the MILP as it can more easily find a good suboptimal solution since many of the solutions are near optimal. This is supported by the visualizations in 5.14a and 5.14b that show that even though the optimizers score similarly in the ground truth metric the schedules produced are different in several areas.
The findings that a timestep of $\approx$ 8-16 hours is beneficial for solution quality is supported by the timestep ablation study in A.4 and by the results in Section 5.1.4. This result is surprising because a higher resolution control should lead to better outcomes. However, the components in the mountain hut does not benefit from rapid oscillatory behavior. Instead the fuel efficiency and the quality of service benefits from having longer periods of undisturbed activation. Longer time steps than 16 hours start to deteriorate in performance as shown in Section A.4 and there are two reasons why this would not be beneficial for an increased ground truth metric. The battery is a limiting factor as time steps get longer because it must be large enough to absorb the energy generated in a single timestep, since a lower temporal resolution leads to larger energy increments. If it cannot do that then the battery cannot be charged and the ground truth metric will be reduced. Another factor is that there is a higher benefit of activation the antennas during the day. This creates higher ground truth metric scores for optimizer runs with time steps that can schedule up time during the day and downtime during the night. Another benefit of using an increased timestep is that it creates fewer decisions, which leads to a smaller chromosome. A smaller chromosome has a smaller solution space and therefore the GA can converge faster and more reliably. A larger timestep also amplifies the penalty signal associated with suboptimal decisions. For example, choosing to activate a lower efficiency fuel cell for 2 hours incurs a relatively small cost, which might be difficult for the GA to distinguish from other near optimal solutions. In contrast, when decisions are made on an 8 hour resolution, the same suboptimal choice accumulates over time, producing a stronger negative signal in the fitness function. This makes it easier for the GA to distinguish between and avoid such decisions, potentially explaining why larger timesteps produce better solutions for the mountain hut.

The results shown in Figure 5.13 illustrate that the MILP produces better energy cost and battery degradation scores. This is surprising as the battery degradation is a nonlinear objective that can be modeled and minimized exactly by the GA and only approximated by the MILP. The better performance of the MILP likely stems from the more efficient trading with the grid, removing the need for the MILP to schedule use of the battery creating a very low usage and in extension a low battery degradation. This is visible in the same figure where the energy cost increases from approximately 5 to 20 units creating a reduction in battery degradation to almost zero.

7
Conclusion

In this section a conclusion of the findings will be presented. A recommendation of which optimizer to chose for which scenario will also be provided.

7.1 Conclusions

- The MILP outperform the GA optimization when there is full knowledge about the renewable energy generation.
- The GA can perform better than MILP in certain cases of weather uncertainties and penalty settings because the MILP creates schedules that operate closer to the battery SOC cut off point. This can be remedied by revising the penalty strategy or applying a rolling horizon optimization.
- The optimality gap increases between the GA and the MILP on the mountain hut scenario with limited initial fuel and increased forecast uncertainty due to the GA becoming stuck in a local optima more easily on more complex optimization problems.
- The optimality gap between the GA and the MILP on the mountain hut scenario shrink for increasing optimization horizons.
- A priority list seeding used on the mountain hut scenario improves solution score by reducing the effect of switching behavior and premature convergence.
- For the mountain hut scenario, a rolling horizon optimization with 24 hour control horizon and a full length planning horizon with a fixed timestep of 8 hours produce better ground truth metric results compared to the same configurations with a variable timestep with a higher temporal resolution the first 24 hours, possibly due to a shorter schedule and more clear penalty signals when larger timesteps are used.
- A timestep between 8-16 improved results for the GA on the mountain hut scenario across forecast lags for the same reasons.
- The MILP performed better than the GA when comparing the tradeoff between energy cost and battery degradation.

7.2 Further improvements

The system models in this thesis are based on linear equations and to improve the simulation accuracy nonlinear models of fuel cells, battery storage and heat pumps could be implemented. Validation on a real system to verify models would also be beneficial to be able to increase system approximation. With a nonlinear simulation

model another comparison between MILP and GA could be done where the GA is tested on the nonlinear models and MILP on the linear models. It could also be interesting to use more advanced metaheuristic models like CMA-ES or other evolution strategies where the search through the solution space is guided differently. GA-MILP hybrids could also be examined where larger systems of components are tested.

7.3 Optimizer recommendations

The MILP generally performs better overall compared to the GA achieving better solution quality for most of the tested cases. For longer time horizons the MILP performs better than the GA but the difference in ground truth metric is low making GA a competitive alternative. The choice of objective weighting for the MILP and penalty configuration for the GA are however important as the schedules produced can be quite different even when the ground truth metric score is similar. The GA also has more hyperparameters to tune making it less user friendly to apply. In the experiments, a Gurobi solver has been used which is a state of the art MILP solver which comes with a licensing cost when used in a commercial setting. Therefore the GA can be a cheaper alternative that can produce near optimal solutions for the mountain hut scenario. Both optimizers can produce schedules that satisfy the temperature constraints on the farm scenario although the GA struggle to correctly optimize energy market interactions, which the MILP has no problem with.

7.4 Limitations

The primary limitation of this study lies in the use of linear component models. While necessary to maintain the tractability of the MILP framework, these approximation overlook nonlinear relationships between inputs and outputs. Consequently, the approximations by the simulation of the real system is reduced, and the results may not capture the absolute global optimum of the physical system. While a MILP model can handle nonlinear component models via non linear programming (NLP) this was not part of the scope of the thesis.

7.5 Implementation complexity

The GA is relatively straightforward to implement at a basic level. There is however large effort going into choosing correct operators, tuning hyperparameters and penalties to achieve good performance. For a fixed problem size the most straightforward way of finding the best hyperparameter would likely be to use a hyperparameter tuning algorithm like grid search or a outer-GA loop encoding the hyperparameters as genes and using the inner GA problem as fitness function.
A further limitation is that the optimal hyperparameter may change even for small changes to the problem, such as adding a component or extending the optimization horizon.

Debugging the GA is however relatively straightforward as each operator acts independently and correct behavior verified by inspecting each step in the process.

Bibliography

[1] Tobias Achterberg and Roland Wunderling. *Mixed Integer Programming: Analyzing 12 Years of Progress*, pages 449–481. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[2] K. Anderson, C. Hansen, W. Holmgren, A. Jensen, M. Mikofski, and A. Driesse. pvlib python: 2023 project update. *Journal of Open Source Software*, 8(92):5994, 2023.

[3] Narges Bidabadi. Using a repair genetic algorithm for solving constrained nonlinear optimization problems. *Journal of Information and Optimization Sciences*, 39(8):1647–1663, 2018.

[4] Carlos A Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11):1245–1287, 2002.

[5] Duc-Cuong Dang and Per Kristian Lehre. Self-adaptation of mutation rates in non-elitist populations. In *Parallel Problem Solving from Nature (PPSN XIV)*, volume 9921 of *Lecture Notes in Computer Science*, pages 803–813, 2016.

[6] Mohammad Reza Ebrahimi and Nima Amjady. Adaptive robust optimization framework for day-ahead microgrid scheduling. *International Journal of Electrical Power Energy Systems*, 107:213–223, 2019.

[7] A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[8] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015.

[9] Clemens Frahnow and Timo Kötzing. Ring migration topology helps bypassing local optima. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1—-8, 2018.

[10] Janice Gaffney, Charles Pearce, and David Green. Binary versus real coding for genetic algorithms: A false dichotomy? *ANZIAM Journal*, 51:347, 06 2010.

[11] Julian Garcia-Guarin, Diego Rodriguez, David Alvarez, Sergio Rivera, Camilo Cortes, Alejandra Guzman, Arturo Bretas, Julio Romero Aguero, and Newton Bretas. Smart microgrids operation considering a variable neighborhood search: The differential evolutionary particle swarm optimization algorithm. *Energies*, 12(16), 2019.

[12] Van-Phuong Ha, Trung-Kien Dao, Ngoc-Yen Pham, and Minh-Hoang Le. A variable-length chromosome genetic algorithm for time-based sensor network schedule optimization. *Sensors*, 21(12):3990, 2021.

[13] Christina Hatzilau, Dionysios Giannakopoulos, A. Thomadakis, S. Karellas, and Emmanuel Kakaras. A review of unit commitment and economic dispatch in power systems. *Journal of Energy Research and Reviews*, 18:23–51, 02 2026.

[14] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, 1975.

[15] Intratec. Methanol price – current & forecasts, 2025. Accessed: 2025-09-01.

[16] Borhan Kazimipour, Xiaodong Li, and A. K. Qin. A review of population initialization techniques for evolutionary algorithms. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2585–2592, 2014.

[17] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[18] Aeidapu Mahesh and Kanwarjit Singh Sandhu. Optimal sizing and energy scheduling of isolated microgrid considering the battery lifetime degradation. *PLOS ONE*, 14(2):e0211642, 2019.

[19] Ajay Maheshwari, Nikolaos G. Paterakis, Massimo Santarelli, and Madeleine Gibescu. Optimizing the operation of energy storage using a non-linear lithium-ion battery degradation model. *Applied Energy*, 261:114360, 2020.

[20] Ali Metiaf, Qianhong Wu, and Yazan Aljeroudi. Searching with direction awareness: Multi-objective genetic algorithm based on angle quantization and crowding distance moga-aqcd. *IEEE Access*, 7:10196–10207, 2019.

[21] Krishnarayalu Movva. Unit commitment with economic dispatch. *International Electrical Engineering Journal (IEEJ) ISSN 2078-2365*, 6:1913–1916, 06 2015.

[22] Amin Nasri, Antonio J. Conejo, Seied Ali Mokari, and Sahand Ghavidel. Unit commitment problem: A new formulation and solution method. *International Journal of Electrical Power & Energy Systems*, 57:236–245, 2014.

[23] Morteza Nazari-Heris et al. Robust optimal scheduling of CHP-based microgrids in presence of wind and photovoltaic generation units: An IGDT approach. *Sustainable Cities and Society*, 74:103191, 2021.

[24] Mohsen Nemati, Martin Braun, and Stefan Tenbohlen. Optimization of unit commitment and economic dispatch in microgrids based on genetic algorithm and mixed integer linear programming. *Applied Energy*, 210:944–963, 2018. Direct GA vs MILP comparison on microgrid UC/economic dispatch.

[25] George L Nemhauser and Laurence A Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

[26] Ajay Raghavan, Paarth Maan, and Ajitha K. B. Shenoy. Optimization of day-ahead energy storage system scheduling in microgrid using genetic algorithm and particle swarm optimization. *IEEE Access*, 8:173068–173078, 2020.

[27] Lars Reinholdt, Jóhannes Kristófersson, Benjamin Zühlsdorf, Brian Elmegaard, Jonas Jensen, Torben Ommen, and Pernille Hartmund Jørgensen. Heat pump COP, part 1: Generalized method for screening of system integration potentials. In *Proceedings of the 13th IIR Gustav Lorentzen Conference on Natural Refrigerants (GL2018)*, volume 2, pages 1097–1104, Valencia, Spain, 2018. International Institute of Refrigeration.

[28] Maël Riou, Florian Dupriez-Robin, Dominique Grondin, Christophe Le Loup, Michel Benne, and Quoc T. Tran. Multi-objective optimization of autonomous microgrids with reliability consideration. *Energies*, 14(15), 2021.

[29] Tomonobu Senjyu, Kai Shimabukuro, Katsumi Uezato, and Toshihisa Funabashi. A fast technique for unit commitment by genetic algorithm based on unit integration. *IFAC Proceedings Volumes*, 36(20):479–484, 2003. 5th IFAC Symposium on Power Plants and Power Systems Control 2003, Seoul, South Korea, 15-19 September 2003.

[30] Martin Serpell and Jim Smith. Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms. *Evolutionary Computation*, 18:491–514, 09 2010.

[31] Vishnu Suresh, Przemyslaw Janik, Michal Jasinski, Josep M. Guerrero, and Zbigniew Leonowicz. Microgrid energy management using metaheuristic optimization algorithms. *Applied Soft Computing*, 134:109981, 2023.

[32] Reza Torkan, Adrian Ilinca, and Masoumeh Ghorbanzadeh. A genetic algorithm optimization approach for smart energy management of microgrids. *Renewable Energy*, 197:852–863, 2022.

[33] Mudith Witharama, Dilshan Bandara, Muhyideen Azeez, Kasun Bandara, Logeeshan Velmanickam, and Chathura Wanigasekara. Advanced genetic algorithm for optimal microgrid scheduling considering solar and load forecasting, battery degradation, and demand response dynamics. *IEEE Access*, 06 2024.

[34] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[35] Bolun Xu, Alexandre Oudalov, Andreas Ulbig, Göran Andersson, and Daniel S. Kirschen. Modeling of lithium-ion battery degradation for cell life assessment. *IEEE Transactions on Smart Grid*, 9(2):1131–1140, 2018.

[36] Jiayi Zhang. Energy management system: The engine for sustainable development and resource optimization. *Highlights in Science, Engineering and Technology*, 76:618–624, 12 2023.

[37] Wen-hua Zhou and Zhen-jie Jiang. Hybrid cataclysmic genetic algorithm used to reactive power optimization. In *2009 Fifth International Joint Conference on INC, IMS and IDC*, pages 1615–1618, 2009.

A

Appendix 1

A.1 GA mean convergence behavior

To know how many times a GA need to be run to be sure that an accurate statistical representation is made a study was done computing the cumulative mean and cumulative standard deviation of 50 GA runs on the mountain hut scenario.

Figure A.1 shows that in order to be sure that the values obtained from a GA experiments are statistically accurate and within a 95 % confidence interval the GA needs to be run $\approx$ 40 times on the same problem.

A.2 GA behavior for adding explicit penalty on switching behavior

The optimized schedules that were produced exhibited oscillating behavior which was thought of as undesirable as the frequent switching of antennas and fuel cells would lead to a decreased quality of service. The oscillating behavior was a result of the goal of having a battery SOC at a fixed point. Therefore a explicit penalty on switching was introduced to investigate whether it would increase the ground truth metric scores.

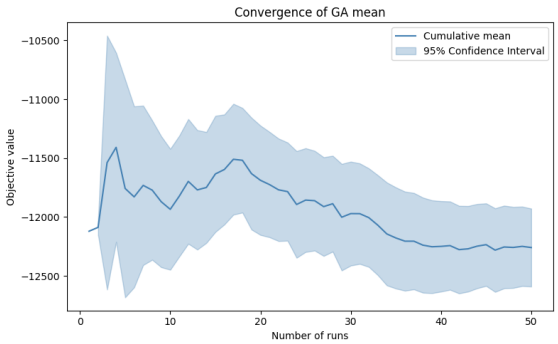
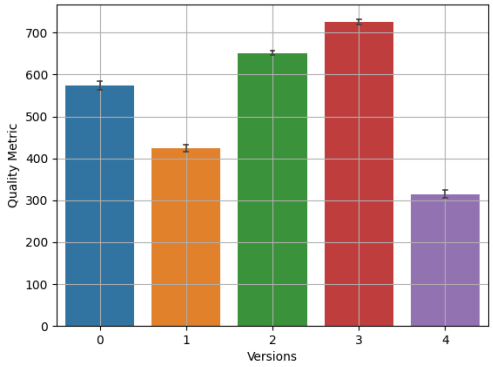
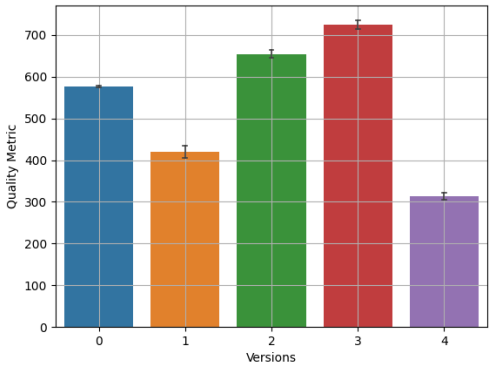


Figure A.1: Convergence of GA mean fitness value on mountain hut scenario with a 95 % confidence interval



(a) Ground truth metric performance across mountain huts with priority list seeding and without a penalty for switching.



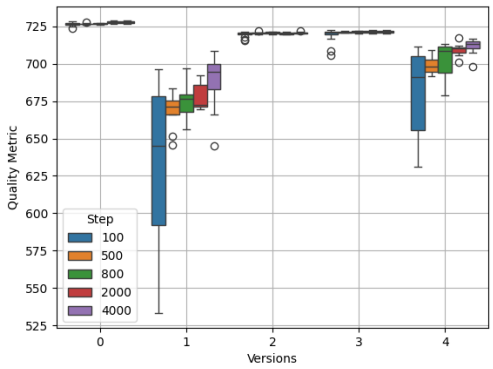
(b) Ground truth metric performance across mountain huts with priority list seeding and with a penalty for switching.

Figure A.2: Switching penalty comparison

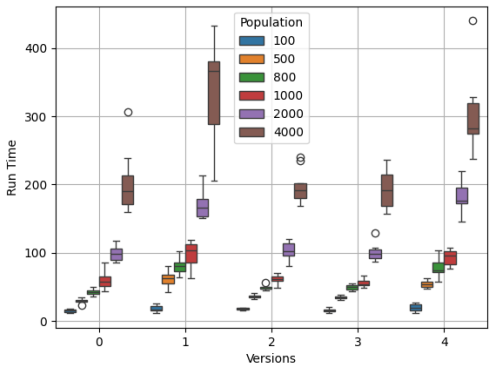
Figures A.2 shows that the adding of an explicit penalty for reducing switching does not effect the performance of the optimizer in any noticeable way.

A.3 Fitness and runtime behavior for increasing population sizes

A larger population size exposes the GA to greater genetic diversity, allowing it to explore more of the solution space and generally improving the final fitness scores. To investigate if the optimality gap between the GA and MILP solution can be reduced further by increasing the population size tests were run across increasing population sizes. Additionally, the effect of population size on computational runtime is verified, as larger populations increase the number of chromosome evaluations per generation.

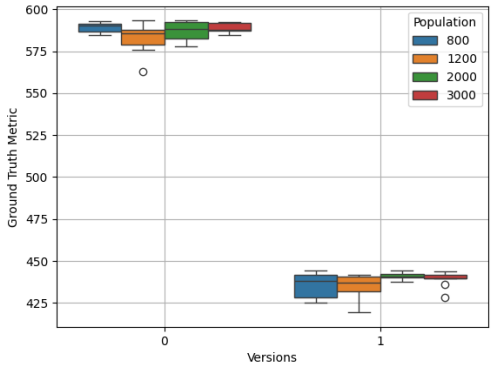


(a) GA ground truth metric performance, complexity 0.

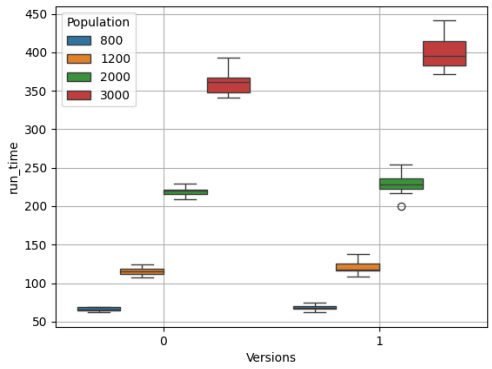


(b) GA run time performance, complexity 0.

Figure A.3: Examination of ground truth metric performance and run time for increasing population sizes.

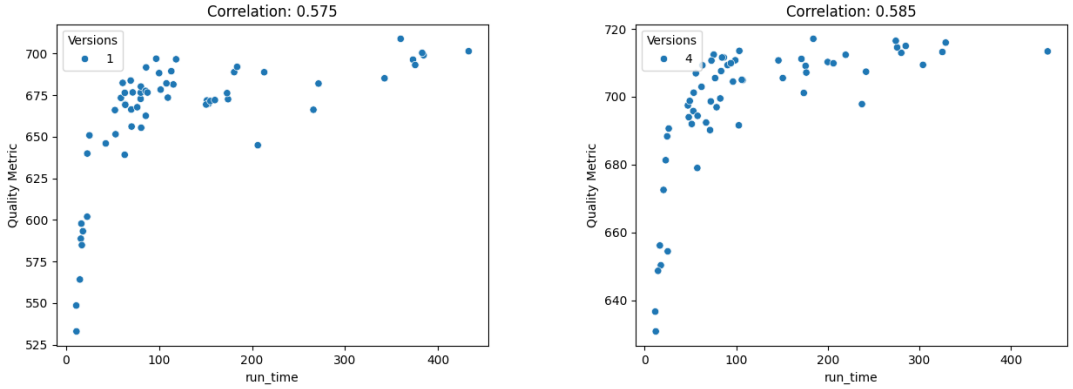


(a) GA ground truth metric performance across mountain hut version 0 and 1. Complexity 2



(b) GA run time performance across population sizes and mountain hut version 0 and 1. Complexity 2.

Figure A.4: Examination of ground truth metric performance and run time for increasing population sizes.



(a) Correlation on fitness vs run time for mountain hut 1 with complexity 0.

(b) Correlation on fitness vs run time for mountain hut 4 with complexity 0.

Figure A.5: Correlation plots of run time and fitness scores. Complexity 0, no forecast error.

Figure A.3a shows that the fitness increases for version 1 and 4 with increasing population size. However, the fitness does not improve much for versions 0, 2 and 3. A reason for this might be that the GA achieves near optimal solutions for version 0, 2 and 3 even with a lower population size. Figure A.4a shows a population ablation for a harder scenario with limited initial fuel. The Figure show that increased runtime in these cases also does not correlate to increased fitness indicating that increasing the population size does not lead to better solutions when the energy availability is limited. Figure A.3a shows that for versions 1 and 4 an increasing population correlates to an increasing performance score. This is further supported by Figure A.5a and Figure A.5b which show that the correlation of fitness to run time is 0.575 and 0.585 respectively.

A.4 Convergence behavior for varying fixed timestep sizes

As shown in Figure A.3b the GA run time increase linearly with the population size. Since runtime also increases linearly with the chromosome length, given that the fitness function evaluation is linear, optimizing over longer horizons or with additional components will eventually become computationally intractable. Therefore it becomes interesting to investigate the effect of increasing the timestep duration, trading temporal resolution for reduced computational cost. To examine the trade off between fitness performance and compute time, several optimization runs were performed with different timestep resolutions.

The plots in this section analyze optimizer behavior across forecast lags and timestep sizes for mountain hut scenario 0, 1 and 2, representing three configurations of the mountain hut scenario with varying difficulty. The plots show behavior across forecast lags and step sizes under a high fuel availability while the later plots show the

constrained scenario with limited fuel.

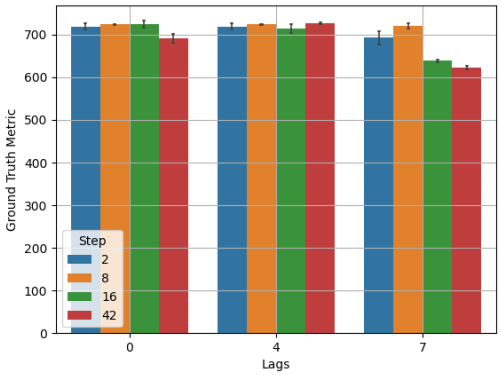


Figure A.6: Ground truth metrics comparison across forecast inaccuracies and timestep frequencies on mountain hut version 0, complexity 0

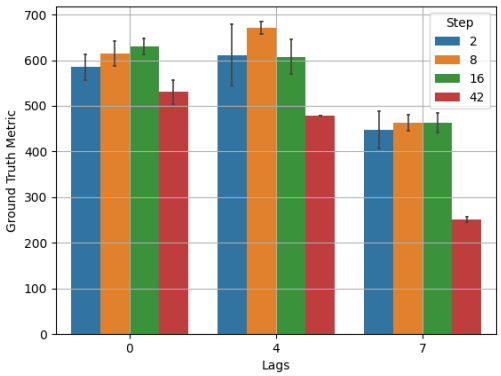


Figure A.7: Ground truth metrics comparison across forecast inaccuracies and timestep frequencies on mountain hut version 1, complexity 0

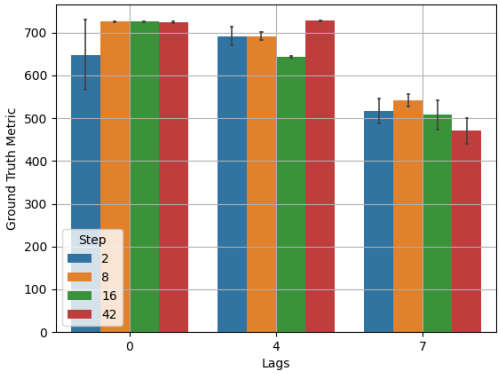


Figure A.8: Ground truth metrics comparison across forecast inaccuracies and timestep frequencies on mountain hut version 2, complexity 0

Figure A.7 shows that a timestep duration of 16 achieves the highest performance when forecasts are reliable, however the performance degrades as forecast lag increases. A time step duration of 8 hours appears more robust across the forecast lags making it a more desirable choice when faced with prediction uncertainty. Notably the time step length with the highest temporal resolution does not perform the best.

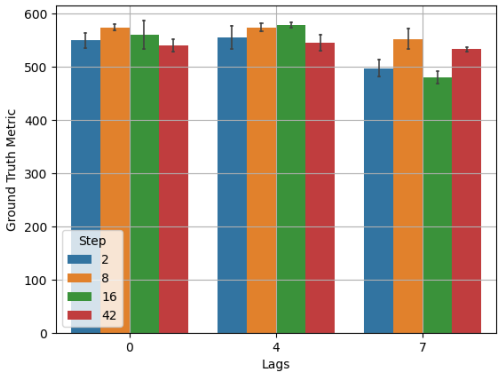


Figure A.9: Ground truth metrics comparison across forecast inaccuracies and timestep frequencies on mountain hut version 0, complexity 2

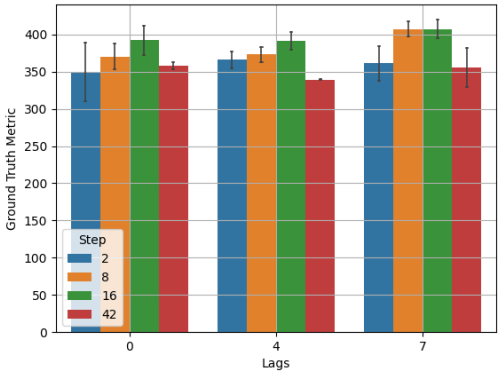


Figure A.10: Ground truth metrics comparison across forecast inaccuracies and timestep frequencies on mountain hut version 1, complexity 2

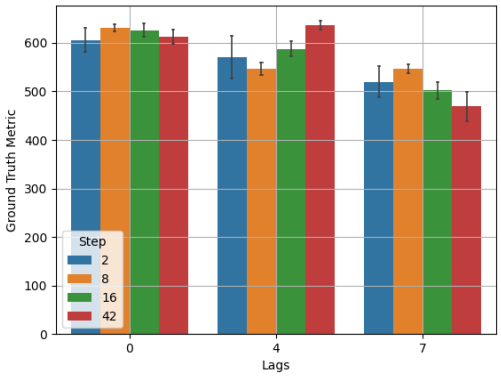


Figure A.11: Ground truth metrics comparison across forecast inaccuracies and timestep frequencies on mountain hut version 1, complexity 2

Figure A.9 , A.10 and A.11 show that with a limited amount of initial fuel available the optimizer still perform better for a larger timestep. Across the forecast errors we can see that in A.9 the timestep frequency of 8 performs the most consistent. Yet in A.10 a timestep duration of 16 performs the best in spite of the version having a small battery capacity.

A.5 Termination criteria

Finding the correct termination criteria was done by comparing the increase in fitness scores for different simulations. If a optimization converges fully across multiple runs after a number of generations that signals that the optimization can be terminated.

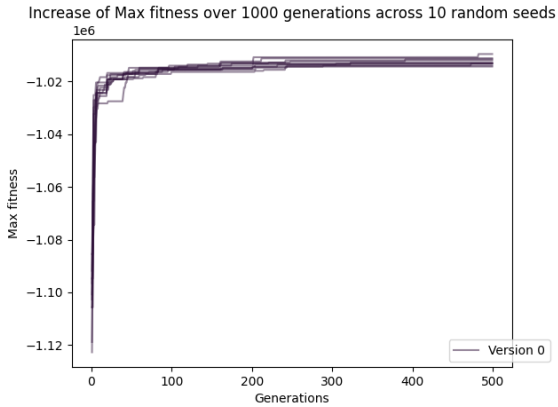


Figure A.12: Max fitness vs generations for mountain hut version 0, 5000 hour simulation period, 10 h time step.

Figure A.12 shows that the fitness values improve rapidly during the initial generations and then flatten out. However, there are still improvements made in fitness for some runs of the GA in the later generations before termination. Figure A.13 shows the convergence behavior for the scenario with 336 hours simulation horizon where

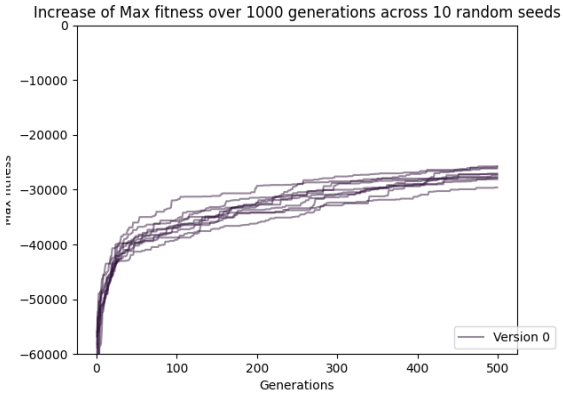


Figure A.13: Max fitness vs generations for mountain hut version 0, 336 hour simulation period, 4 hour time step.

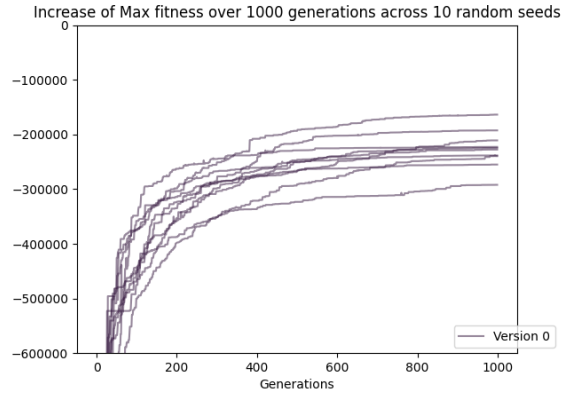


Figure A.14: Max fitness vs generations for farm scenario version 0, 48 hour simulation period, 1 hour time step.

the scores also improve fast initially and then flatten out. It is clear however that improvements in the fitness score are still being made in the later generations. Figure A.14 show a similar convergence behavior as Figure A.13 where improvements to the fitness score continue in the later generations. These figures illustrate that most of the convergence happen in the early generations but for all experiments improvements are made close to the termination border, with Figure A.12 showing most convergence.

B

Appendix 2

B.1 Component equations

Battery update rule

$$\mathrm{SOC}_b(t) = \mathrm{SOC}_b(t-1)\Big(1-\sigma_b(t)\Big) + \frac{r_s(t-1)}{E_s}\,\eta_s^{\mathrm{ch}} - \frac{r_s(t-1)}{E_s\,\eta_s^{\mathrm{dis}}}$$

B.1.1 Battery degradation model

The battery degradation was calculated as an accumulation of the fraction of the battery life that was consumed

$$\text{Battery degradation} = \sum_{i}^{I} \frac{\gamma_i}{L(\xi_i)} \tag{B.1}$$

Where γ_i is the number of discharges at a specific DOD given by the rain flow counting, ξ_i is the DOD value and I is the number of bins used to section the DODs. $L(\xi_i)$ describes the number of discharge cycles a battery can withstand given a specific DOD and is described as

$$L(\xi) = a\xi + b$$

where a = -22467 and b = 2873 are empirically derived parameters fit to manufacturer cycle life data [35]. This model only accounts for cycle aging of the battery and disregards other reasons that might cause degradation in the battery.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY