Karlsruhe Institute of Technology

A Multi-Objective Optimization Approach for Bulk Material Blending Systems

Diplomarbeit
von

Michael P. Cipold

am Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB)
der Fakultät für Wirtschaftswissenschaften

Referent:	Prof. Dr. Hartmut Schmeck
Betreuer:	Dr. Pradyumn Kumar Shukla

Bearbeitungszeit: November 5, 2012 – May 3, 2013

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person, except where due acknowledgment has been made in the text.

Karlsruhe, May 3, 2013

Acknowledgments

I would like to express my gratitude to Dr. Pradyumn Shukla who supported me during the whole research period and the development of this thesis as a thoughtful and patient mentor. I am especially thankful for giving me the opportunity to present the findings of this thesis on a leading conference on evolutionary optimization.

Furthermore I would like to thank Dr. Claus Bachmann for his permanent encouragement and support of the development in new fields of technology and interesting research projects which made the investigation of this extraordinary topic possible.

My thanks are also directed to Prof. Dr. Hartmut Schmeck for the opportunity to write this thesis at the AIFB.

In the end I would like to thank everyone else who supported me with advice, information and especially with patience in times of hurry.

Abstract

Nowadays bulk material blending systems still mainly implement the static well-known chevron stacking. Real-time quality measurement techniques as online X-ray fluorescence measurement allow the dynamic adaptation of the blending process to the current quality data. We propose a multi-objective optimization system on the base of steady state evolutionary algorithms using various Baldwinian and Lamarckian repair algorithms and test the algorithms on real world problem data. The optimized solutions generally outperform the standard techniques and furthermore allow insights and deeper understanding of the problem nature and the algorithms presented.

Kurzzusammenfassung

Heutzutage verwenden Schüttgut-Mischbettsysteme immernoch hauptsächlich statische Methoden zum Aufbau der Mischhalden, wie das bekannte Chevron-Stacking. Die Echtzeitmessung der Materialqualität, wie beispielsweise mit Hilfe der Online-Röntgenfluoreszenzmessung möglich, erlaubt die dynamische Anpassung des Mischprozesses an die aktuelle Qualität. Diese Arbeit präsentiert ein Optimierungssystem für mehrere Zielparameter basierend auf verschiedenen Baldwinischen und Lamarckschen Reparaturalgorithmen und zeigt die Funktionalität anhand der Daten aus einem tatsächlichen System. Die optimierten Lösungen übertreffen immer die Lösungen, die mit statischen Methoden berechnet wurden und erlauben weiterhin Einsichten in die Natur des gestellten Problems und ein tieferes Verständnis der vorgestellten Algorithmen.

Contents

1. Introduction

This thesis presents research in bulk material blending system optimization using a multi-objective problem model.

1.1 Bulk Material Processing

Bulk material processing describes the handling of raw bulk materials such as minerals, coals or ores. These are mostly mined from natural source and thus are usually inhomogeneous in matters of quality parameters. Quality equalization to guarantee efficient processing can be achieved by using blending systems in the processing chain. Most systems today still implement static blending methods like the well-known Chevron stacking. Blending with static methods results in partially equalized quality output curves which still contain fluctuations exceeding the desired limits in many cases. Real-time quality measurement makes it possible to follow the current quality trend and react dynamically with the blending process.

This thesis utilizes a particle simulation to reproduce the blending process virtually and allows calculation of the outcome of a possible solution during optimization. The optimization is done modifying the stacking machine traverse path only which implements a method of minimal system change and thus with minimal financial and hardware effort.

1.2 Multi-Objective Optimization

The optimization has to meet the main objective of minimizing the quality variation of the material leaving the blending system. But the system is bound with several other limits, for example maximum speed of the stacking machine or stockpile limits. The machines additionally limit the shape of the stockpile; they require the height to be nearly constant. As many real world problems the objectives describe conflicting goals and result in a multi-objective optimization problem.

This thesis provides a calculation model and methods to produce feasible optimized solutions for the stacker traverse path regarding the objectives of choice. Solutions produced by the optimization system always outperform results which are achievable by static blending.

The analysis of solutions found by the optimization system allows insights about common principles of good strategies and thus Innovization [Deb01]. Furthermore the analysis allows the derivation of knowledge and understanding about the algorithms used for optimization.

1.3 Related Work

The field of bulk material blending optimization is not new to the research world. Many authors have already analyzed and improved the blending efficiency but the ways described in this thesis have, to the best of our knowledge, never been presented by others in research publications.

Kumral [Kum03] describe a method to optimize a mineral blending system design before its construction to get the best performance. Using genetic algorithms and a multiple regression model they determine the best blending bed parameters simulating the stockpile with a simplified cell model.

Pavloudakis and Agioutanis [PA06] use a more complex stockpile simulation approach consisting of multiple layers lying on top of each other. They assume a constant material flow and calculate the expected quality at the material output. It is not scope of their work to optimize the blending efficiency.

Bond et al. [BCW00] describe methods to improve the efficiency of blending beds by modifying the volume of a stockpile dynamically and modifying the stacking speed while measuring the input quality with an online analyzer. They only suggest a solution with an *appropriate software* to control the stacker speed dynamically during the stacking to place material with recognized quality at specific locations in the blending bed but do not extend this thought in detail.

The particle based simulation and the evolutionary multi-objective optimization approach make the research presented in this thesis unique. They offer a flexible system to use in many kinds of blending simulation and optimization problems not only with the blending bed types and machines presented in this paper.

The findings of the research were partially published at the *International Conference on Parallel Problem Solving From Nature 2012* [CSB$^+$12] and were extended with new algorithms, results and insights.

1.4 Structure of the Thesis

This thesis consists of three main chapters. Chapter 3 describes the problem modeling with its objectives, constraints, requirements and the simulation approach. Chapter 4 presents the solution for the optimization problem with its algorithms, mechanisms and operators. Chapter 5 shows results achieved using the presented techniques and proves the suitability of the methods. Furthermore insights about algorithms and solutions are concluded from the data which was collected during the tests. The final chapter 6 summarizes the conclusions and illustrates the next steps to be done regarding this research topic.

2. Fundamentals of Bulk Material Processing

This chapter gives a quick introduction in the basic knowledge of bulk material processing and related topics. This knowledge is necessary to motivate and understand the need for the optimization described in chapter 3.

2.1 Bulk Material Processing

Bulk material processing in sense of this thesis describes the handling of raw bulk materials such as minerals, coals or ores. These materials are mostly mined from natural resources and hence the product is usually inhomogeneous in matters of the quality parameters of interest. This may be illustrated for coal. Here typical quality parameters are moisture or ash content as not burning components, sulfur content or calorific value and lots more [Ald12].

In most bulk material processes a range of acceptable quality parameters can be defined for the process to work properly. The process is then adapted for the specified average quality. As the quality of the material varies over time the process is not operated in the most efficient way and the used process equipment can even take damage from too high variation even though the material meets the average quality setpoints. As an example an autoclave for coal gasification may be considered. This unit should be run at a specific heat setpoint in order to operate most efficiently. Even though the average particle size is constant during the day some minutes or even hours of smaller particles (balanced by bigger particles in the later time of the day) can cause the oven to heat up over the target setpoint. This is critical for the machines used and has to be detected, the material flow has to be reduced and the process is influenced. Even more critical situations occur if the calorific value is not longer balanced to the amount of ash in the coal. If the energy contained in the coal is not sufficient to keep the temperature above a certain limit the fluid ash will solidify which puts the autoclave out of service.

Summarized it can be stated that the absolute average quality value is not easily controllable but also the control is not necessary as long as the process is adapted

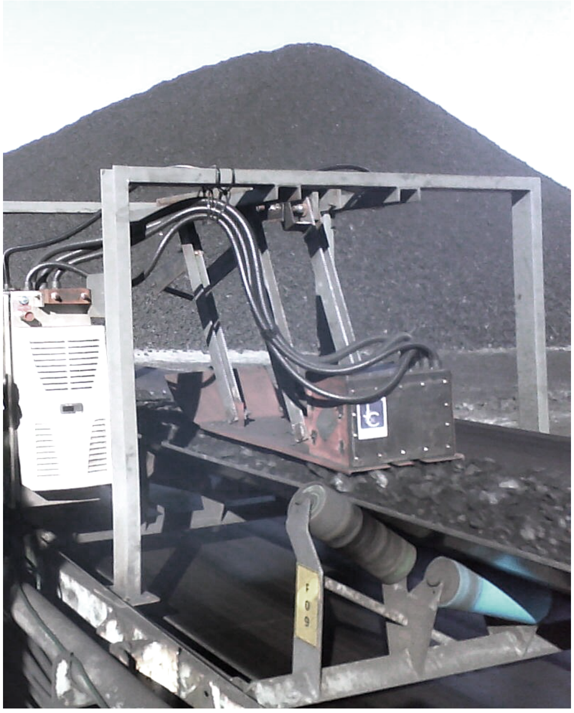


Figure 2.1: An online X-ray fluorescence analyzer mounted on a sled gliding over coal on a conveyor belt

for the specified quality value but there is a requirement for a stable quality which means a low quality variance.

2.2 Quality Measurement

As described in section 2.1 many parameters can be of interest in bulk material processing. The parameters are usually defined by measurement instruments in laboratories after taking samples of the material stream.

In the last decade a whole range of instruments has been developed to allow online determination of quality parameters like elemental composition, size distribution and other parameters of interest. These instruments are determining the quality parameters directly on the main stream (hence the word online) and allow the operator to create a regulated process which is adapted to the current parameter measurement. The main advantage of these instruments is the lack of the necessity to draw samples and the real-time quality determination.

The measurement instrument used to determine the quality parameters used for the data in this thesis is an online X-ray fluorescence analyzer [LB04]. The instrument is located over or even on the main material stream as it can be seen in figure 2.1. An X-ray beam is directed onto the material surface and fluorescence radiation can be observed by a silicon detector. The energy resolved spectra are then analyzed and the elemental composition and other desired process parameters (like moisture) can be derived from the spectra's results. The instrument updates the current quality values every couple of seconds and is therefore usable for real-time control.

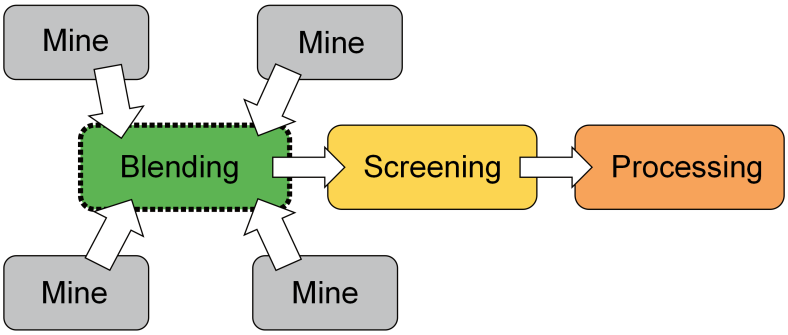


Figure 2.2: Illustration of direct mining and processing

2.3 Bulk Material Blending

Bulk materials are typically originated by a mine. These natural deposits are typically inhomogeneous in quality and therefore in most cases the material needs to be processed in a plant.

Bulk material processing usually requires the quality trend of the input material to have a low variation as described in section 2.1. Bulk material is mined at places where exploration results indicate a worthwhile material extraction and then it is normally directly transported to the respective processing plant after some minor handling as illustrated in figure 2.2.

The mining pattern limits the average quality and assures it does not exceed the given limitations. But as layers of material are mined the quality varies strongly. Some variation levels can last for hours and even for days. To minimize the negative effect which poor material would have onto the process blending is inserted before the processing. The aim of the blending is a homogenization of the raw material to minimize the quality variation.

This thesis is focused on longitudinal blending beds as illustrated in figure 2.3. The blending bed is an area allocated for building so-called stockpiles of raw material. The blending bed can have a typical length up to 1000 m and a width up to 50 m. The bed is located between two rail tracks on which huge stacking and reclaiming machines move along the bed which can be seen in the figures 2.5 and 2.4 published on [sta13, rec13]. The stacking machine is fed with the input bulk stream using a conveyor belt. The reclaimer collects the bulk material on one end of the stockpile and feeds it to a conveyor belt moving to the plant. The stacking machine has a long arm which reaches over the middle of the blending bed where the material is deposited. The material is then stacked to stockpiles by layering it with so-called cone-shell or chevron stacking methods [BCW00]. For instance using the chevron stacking method the material is layered in horizontal levels. Defined by the speed of the stacking machine and the material flow the levels become thicker or thinner. The bridge reclaiming machine ablates the material in diagonal layers. If the stockpile was stacked using the chevron stacking method the cross sections ablated diagonally contain material of all levels. The blending result correlates mostly with the amount of layers and hence with the speed of the stacking machine but is also influenced byte the material quality curve shape as illustrated in detail in section 5.4.

As standard the necessary amount of layers for a good blending result is calculated statically and optimized mainly at design time together with the general blending

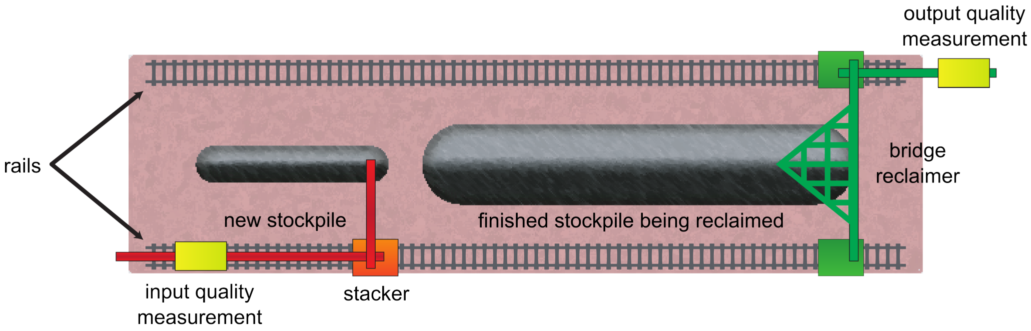


Figure 2.3: Illustration of a longitudinal blending bed with railed stacking and reclaiming machines

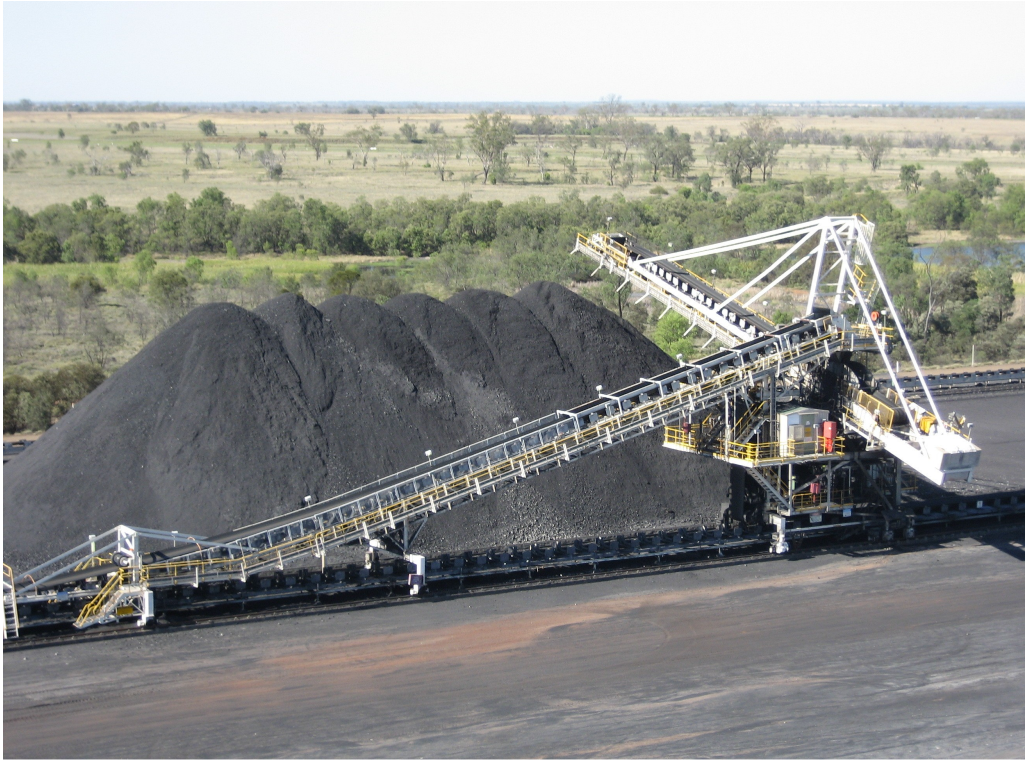


Figure 2.4: Photo of a railed stacking machine at a longitudinal blending bed

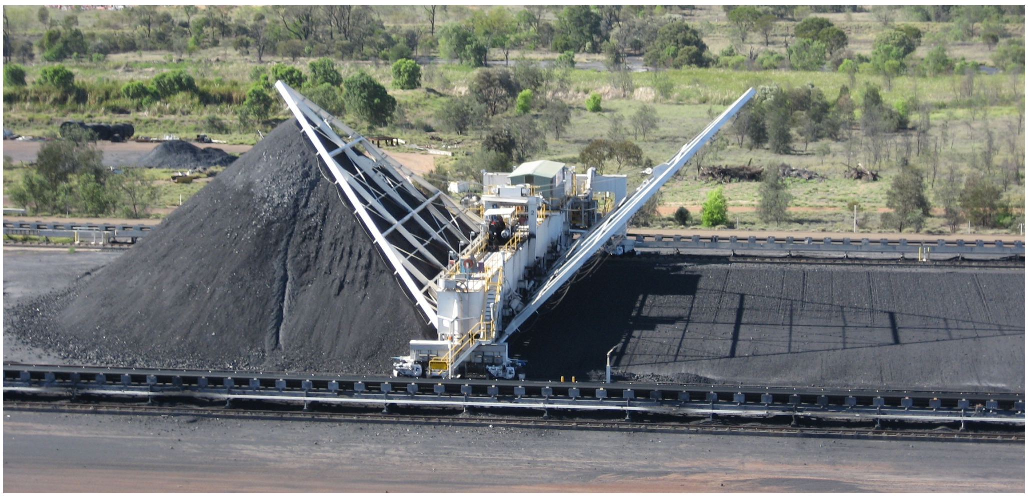


Figure 2.5: Photo on a railed reclaiming machine at a longitudinal blending bed

bed parameters [Kum03]. In some cases the amount of layers is varied by experienced operation personnel but not directly controlled in reaction to quality variation changes.

Though this thesis focuses on longitudinal blending beds with the machines as described chapter 6 gives a short introduction how the principle can be extended to other kinds of blending bed types like for instance circular blending beds [Pet04].

3. Modeling and Simulation

This chapter describes the problem and its modeling in detail which is the base for the algorithms presented in the next chapter. The variables, objectives, constraints are specified together with the detailed problem description and the requirements for the solution model. Furthermore a simulation system is described for evaluation of possible solutions.

3.1 Problem Description

As stated in section 2.1 bulk material processing requires a low quality variance in the material stream. To achieve this blending beds are inserted into the processing chain to buffer good and bad materials and homogenize them into a usable product.

The stacking machines driving along the blending bed are controlled statically and place a fixed amount of layers on top of each other while going back and forth periodically along the current stockpile length. This process results in non-optimal homogenization which does not always satisfy the process needs as illustrated in section 5.5.

With the current measurement and control potential described in section 2.2 the options are available to influence the blending process reactive to the current material quality. The manipulation of the stacking machine traverse path offers the possibility to optimize the homogenization result and minimize the quality variance further as to the current degree. The manipulation of the traverse path can be done using the existing technology upgraded with optimization software with thus with low financial and hardware effort. The traverse path has to meet several constraints of which some are hard constraints and others can be formulated as optimization objectives. This results in the traverse path being the main variable in a multi-objective optimization system as described in detail in section 3.2.

The following chapters present a possible model the for the muli-objective optimization problem and algorithms to solve the problem.

3.2 Optimization Variables

As the section 3.1 states the optimization of a bulk material blending system with minimal hardware and financial effort with fixed system parameters like blending bed size, stockpile dimensions, machine maximum speed and the dynamic material quality curve given the only variable left for optimization is the stacking machine traverse path.

The modeling of the machine traverse path is part of the algorithm description and can be found in section 4.1.

3.3 Optimization Objectives

Section 3.1 describes the main goal for the optimization of the bulk material blending system. The blending is optimized when the material output curve has a minimum variance and thus is close to the mean value. This results in the output quality variance being the first main objective described formally in section 4.3.

An optimized solution has to meet several hard constraints which are described in the following section. But one additional requirement to a possible stacking machine traverse path is the nearly constant height of the resulting stockpile. It has to stay within the limits given by the machines and the blending bed. This requirement could be formulated as a hard constraint but within this thesis this height constancy is selected as the second main objective to be optimized. The reason for this is the partially available tolerance for height differences in the resulting stockpile. Certain degrees of height differences can be tolerated. The even more important reason for including the height difference as a second objective is the research of solutions which are given a higher degree of freedom building a stockpile. This way solutions can be found with not as constant height but with lower quality variance and solutions which are strictly restricted in height difference and their achievable quality variance values. Again the formal description of this objective is part of section 4.3.

3.4 Constraints

A feasible solution for a stacking traverse path has to meet several constraints. The stacker may not exceed a given maximum speed and also it may not exceed the borders given by the stockpile definition. These two constraints are usually violated by the solutions calculated by the optimization algorithm presented in chapter 4. To create solutions which are still feasible certain repair mechanisms will be applied.

Another constraint is the stockpile itself which may also not exceed the limits given by the defined limits. This constraint is mainly handled by the second objective described in the last section and the limited traverse region. If the height difference within the stockpile region does not exceed certain values and the traverse path is validly within the borders and the amount of material was calculated properly the stockpile will satisfy the constraints.

3.5 Requirements

A main assumption has to be made for the optimization system to work. The optimization system described in this thesis calculates an optimal solution based on the knowledge of the full quality input curve. If the quality measurement was located directly in front of the stacking system as shown in figure 2.3 this assumption would not allow a real world optimization and would not be of practical value. But the fact that quality measurement can be installed at the material source meaning at the mines and the second fact that usually there is more buffering between the mine and the processing plan (e.g. silos) the quality curve for a real world optimization system can be predicted in high detail. The choice of material from different mines can be predicted, too or even influenced, the buffering allows detailed prediction over the future development and the stability of quality levels coming on the material stream from a specific mine. Also the routes of transport are quite long and the material measured at the mine arrives at the processing plant many minutes to several hours later. Summarized this means a good prediction of the quality curve can be made which can be used for optimization.

3.6 Quality Data

The quality data recorded by the measurement system described in section 2.2 was collected in a coal processing plant in South Africa. At the measuring point coal of up to seven different mines is fed to the stockpile. The measuring device can be calibrated to deliver elemental composition but also other material parameters like moisture or ash content. As stated in section 2.1 the quality of a raw material can be composed of many values. To keep the explanations of the methods used in this thesis easily understandable the abstract description *quality* is used to describe the material. The actual data used in the quality input curves for the optimization was the ash content of the coal.

Though this radical simplification was made the methods described are usable in general as the extension of the optimization to more parameters is only a matter of effort but not a change to the algorithms themselves.

Furthermore there is a high (positive or negative) correlation between the single quality parameters which explains why the optimal distribution of one quality parameter usually also results in a good distribution of many other parameters. This is due to the fact that the quality usually varies strongly as the material stream changes its source to another mine or one mine mines from a different spot with a different material composition in which itself the quality is relatively constant again.

Figure 3.1 shows a typical quality curve. As the data recorded in the coal processing plant is kept under corporate secret only a small window of actual data could be used and published within this thesis. Further data was generated based on the real world data in order to test and present the optimization system with other quality curves with different properties. The generated data has the same basic properties as the original data but varies in amount, size and stability of quality levels. All used quality curves are visualized in figure 5.1 on page 28.

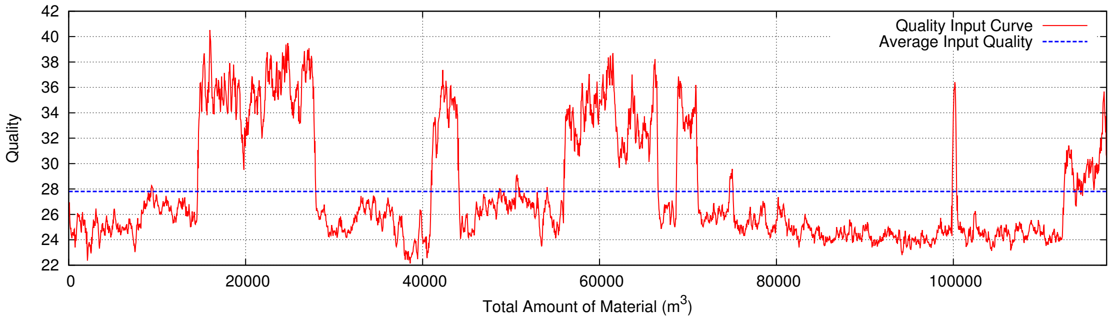


Figure 3.1: Typical quality input curve

3.6.1 Quality Data Model

To model the quality curves a normalization was done in the first place in order to have an easily manageable model. The actual quality curves were recorded over time; additionally the current material flow was tracked. Sometimes the material flow stopped completely, sometimes the belt was only partly loaded. In order to make two quality curves comparable a representation was chosen where the current material quality was mapped to the current total amount of material stacked and respectively ablated regarding the reclaiming machine.

As a model representation a simple floating point value array of qualities q_i was chosen where each entry represents the average quality for a specific, fixed amount of material as illustrated in equation 3.1. All quality slots together form the specific quality curve $\mathbf{q}$.

$$\mathbf{q} = (q_1, \ldots, q_{l_q}), q_i \in \mathbb{R}, l_q \geq 1 \tag{3.1}$$

3.7 Blending Bed Simulation

As potential solutions for the blending bed traverse path optimization problem can not all be tested in the real world for reasons of cost, time and implementation limits a simulation system is required to allow evaluation of potential solutions. This simulation based on a given quality input curve has to produce an output with high correlation to the output the real world system would have.

In a first step during preliminary work for this thesis a detailed physics simulation system was implemented. The detailed simulation was adapted to represent the real world as close as possible by verification with a number of tests and review by experienced bulk material processing personnel. The details for the implementation of the detailed simulation an be found in section 3.7.1

As the detailed simulation takes roughly 20 hours on present day personal computers with the parameters used for the results presented in section 5 a direct optimization with this simulator is only possible on so-called super computers. To run the optimization on a personal computer a different, fast simulation system has been developed during further preliminary work. This fast simulation takes roughly 15 milliseconds to simulate a full stacking and reclaiming process. To achieve this huge speedup several simplifications were made to the physical assumptions as described in detail in section 3.7.2

The quality output curves of both simulation systems still show a high correlation as shown in section 3.7.3. Therefore the fast simulator can be used to calculate an approximate result for a potential optimization solution.

Both simulation systems are parametrized in the same way: the static parameters (blending bed size, simulation detail) are set at the beginning of the simulation. Then the input quality curve is fed as a stream of quality values normalized on the current total amount of material together with the current traverse path position. The simulation systems build up the stockpile by simulating single particles with the given quality deposited by the stacker. Finally the stockpile is ablated with the reclaimer and the quality output curve is generated. This quality output curve is again normalized on the current total amount of material.

The particle based simulation is one key difference to other works in the field of blending bed optimization. It offers a highly detailed and reliable simulation of the stockpile with various parameter combinations where most stockpile models have to be adapted manually for each case.

3.7.1 Detailed Simulation

The detailed simulation of the blending bed was implemented using a simulated stacker throwing particles in a parabola in direction of the simulated conveyor belt. Each cuboid particle parametrized with specific material parameters for friction, size and weight also carries the information of its quality. The simulation of the physics is done with the Bullet Physics Library [bul13] using standard collision detection of rigid bodies.

The particles fall onto a simulated ground where they create small heaps which grow bigger and bigger the more particles fall on. New particles slipper or roll down the sides as they do in the real world. As a result a full stockpile is created. This stockpile consists of all the particles which were simulated each assigned with a specific quality. Finally a bridge reclaimer is simulated which swipes diagonally through the stockpile calculating the average quality for each slice.

Figure 3.2 shows a partially simulated stockpile buildup process with color coded particle boundaries indicating the quality of each specific particle and an average height map in gray.

When the detailed simulation software was implemented the physics simulation was taking about 20 hours of simulation time for building one stockpile simulated on one CPU only. Meanwhile the Bullet Physics Library has evolved [Cou13] and with the use of present day GPU accelerated calculation the simulation could be rewritten and and the simulation time could be shortened drastically. This improvement is not part of this thesis and will therefore not be described and should rather be regarded as potential future work as the speed-up could even make a real physics simulation possible for the direct verification of the optimization candidates.

3.7.2 Fast Simulation

During preliminary work for this thesis a fast simulation system was developed and implemented to allow a quick but realistic assessment of a possible optimization

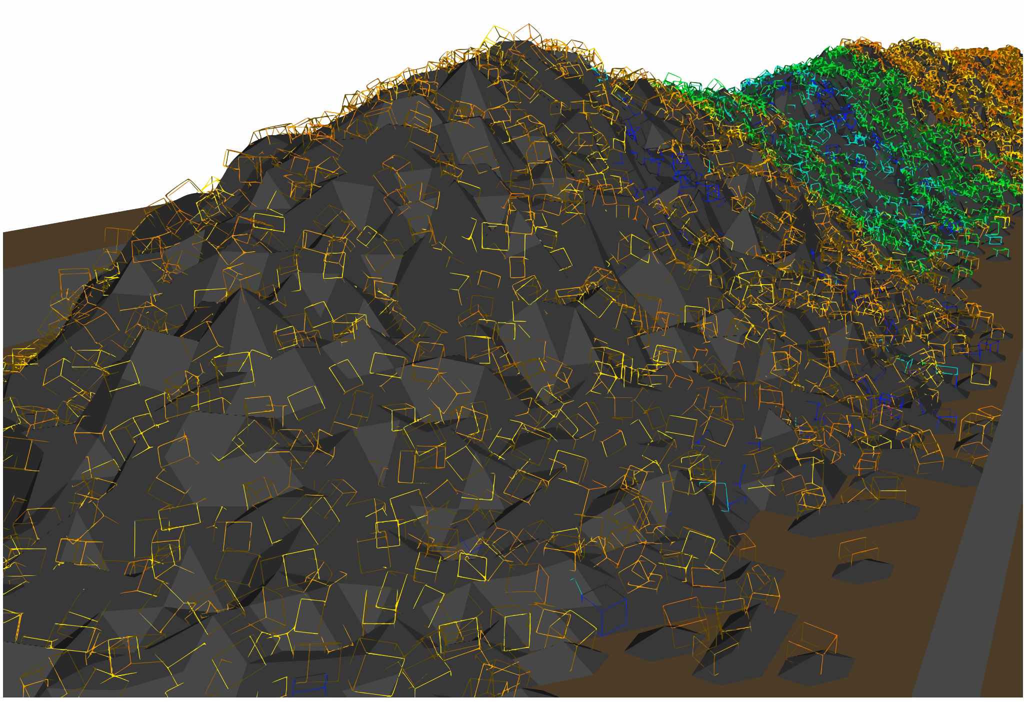


Figure 3.2: A stockpile build and visualized by the detailed simulation

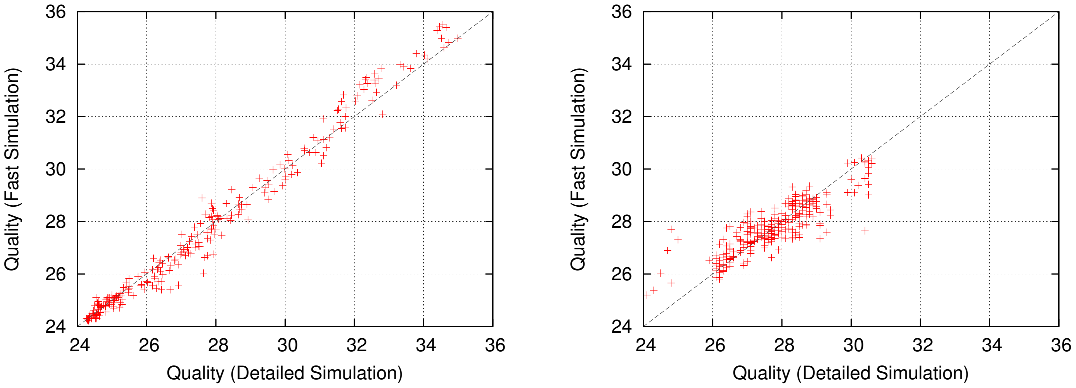
solution. This system implements many simplifications to the physics simulation but still simulates particle based stockpile building.

The stockpile is now quantized into cells in three dimensions. A particle is dropped at a quantized position in the three dimensional space. The grid is checked at this specific position for its height. In the next step the surrounding cells are checked for calculation of the heap the slope. If the slope exceeds 45 degrees then the particle moves to this direction and the sliding check restarts. If there is no direction in which the particle could possible fall the particle stays at its position. The quantization allows all calculation except for the average quality calculation to be done with few integer values instead of floating point position handling and complex body collision detection. Furthermore a lower amount of particles is simulated and all particles are assumed as equally sized. These simplifications make the simulation run in 15 ms on the same computer which simulates the detailed simulation in 20 hours and thus allow the giant speed-up and the use in the optimization.

Though the simplifications used to achieve the huge speed-up are radical the simulation result shows a high correlation with the simulation result of the detailed simulator as shown in figure 3.3 in section 3.7.3.

3.7.3 Comparison

In the last sections the detailed and the fast simulations were presented. To prove the suitability of the fast simulation for the optimization figure 3.3 illustrates the high correlation between the quality output curves of the detailed and the fast simulation. The Pearson correlation for the shown high quality variance is 0.99 and for the low quality variance it is 0.85. Other simulation runs achieved similar results.



(a) High quality variance

(b) Low quality variance

Figure 3.3: Scatter plots showing output quality correlation for detailed simulator versus simplified simulator

3.8 Summary

The presented problem details describe the requirements for an optimization algorithm. Furthermore constraints and objectives are modeled and a simulation system is described for evaluation of possible solutions generated with by an optimization algorithm.

4. Multi-Objective Algorithms

This chapter describes the algorithms used in this thesis for optimizing the multi-objective problem described. Further model definitions are given to specify the representation of individuals. Also possible handling of infeasible solutions is presented as well as the optimization algorithm itself.

4.1 Traverse Path Modeling

To present optimization mechanisms and their effects on the stacker traverse path along the bed the traverse path has to be modeled. Obviously the traverse path of a real world system has to meet the requirements described in section 3.5.

The model can not describe the full movement in complete detail but has to make simplifying assumptions. Hereafter two traverse path representation models are presented which give different optimization possibilities and have different advantages and disadvantages.

4.1.1 Velocity Representation Model

The velocity representation model presented in this thesis describes the stacker traverse path with an array of single velocity values v_i for each specific time slot i of length t^v_{slot} as shown in equation 4.1. The value corresponds to the signed driving speed of the stacking machine relative to the maximum speed. A value of 1 describes the movement with maximum speed into the positive direction (let it be "to the right") and a value of -1 describes the movement with maximum speed into the opposite, negative direction. A value of 0 represents no movement and all values between are interpolated linearly.

$$\mathbf{v} = (v_1, \ldots, v_{l^v}), v_i \in [-1, 1], l^v \geq 1 \tag{4.1}$$

The length of a time slot t^v_{slot} is defined by the amount of time which is needed for all material to be stacked to fill the stockpile t_{total} divided by the length of the array l^v as shown in equation 4.2.

$$t^v_{\text{slot}} = \frac{t_{\text{total}}}{l^v} \tag{4.2}$$

4.1.2 Position Representation Model

This model is based on position sampling and thus utilizes an array of floating point values $p_i \in [0, 1]$ which correspond to the relative position in the valid traverse range of the stacking machine regarding the current stockpile. The definition is shown in equation 4.3.

$$\mathbf{p} = (p_1, \ldots, p_{l^p}), p_i \in [0, 1], l^p > 1 \tag{4.3}$$

The time t^p_{slot} for driving between two positions linearly with constant velocity is calculated as described in equation 4.4 with t_{total} again being the time to fully fill the stockpile.

$$t^p_{\text{slot}} = \frac{t_{\text{total}}}{l^p - 1} \tag{4.4}$$

4.1.3 Comparison of Representation Models

One difference between the two representation models is the fact that the starting position for the velocity representation model needs to be set separately. But for example limiting the starting position representation model to start always at the same position p_1 as the velocity representation model and setting the length $l^v = l^p - 1$ the same set of solutions can be described as long as the solutions are feasible. A possible conversion from one model into the other can be defined as shown in equation 4.5.

$$v_i = \frac{p_{i+1} - p_i}{t^p_{\text{slot}}} \tag{4.5}$$

Still there are differences for the optimization flexibility of these systems and also for the effects of repairing methods as described in section 4.5.1.

One main difference is the effect which changing one value in the array has on the whole traverse path. In case of the position representation model the change of one value affects the movement in the time slot before and after the position. In case of the model of velocities the change of one value affects the whole traverse path from the point in time where the change was made as the path is defined over the integration of the velocities over the time.

4.2 Repair Mechanisms

As it is easy to see the models described to represent the traverse path have their deficiencies. The one which is of interest in this chapter is that not only feasible solutions are representable by the models which makes optimization harder as constraints have to be met which can only be measured after generating a maybe infeasible solution. To lower or even eliminate the amount of infeasible solutions this sections describes repair mechanisms which take infeasible solutions and convert or interpret them in a way that they can be used as feasible solutions again.

4.2.1 Velocity Representation

Calculating the actual traverse path for a solution in the velocity representation obviously results in most individuals being infeasible because the calculated path exceeds the limits of the stockpile. Therefore a simple repairing mechanism was defined using a mirroring technique. As the integration of the velocity over time results in the absolute position p_{abs} the position can be folded into the valid range to $p_{\text{mirrored}} \in [0, 1]$ using equation 4.6:

$$p_{\text{mirrored}} = \begin{cases} p_{\text{abs}} - \lfloor p_{\text{abs}} \rfloor, & \text{if } \lfloor p_{\text{abs}} \rfloor \bmod 2 = 0, \\ 1.0 - (p_{\text{abs}} - \lfloor p_{\text{abs}} \rfloor), & \text{if } \lfloor p_{\text{abs}} \rfloor \bmod 2 = 1. \end{cases} \tag{4.6}$$

This way all positions exceeding the range $[0, 1]$ will be mirrored at the limits into the valid range. The repaired representation can not be mapped back to an individual because there are additional changes in the direction between two regular velocity changes in the array model. Mapping back the mirroring means changing the amount of variables and the fixed time between two velocity changes to be set per velocity.

Example 4.1. Let

$$v_{\max} = \frac{2 \cdot width_{\text{stockpile}}}{t_{\text{total}}}$$

be the maximum stacker traverse speed which represents the speed to place in maximum two layers of the full width of the stockpile traverse path $width_{\text{stockpile}}$ in the total time available t_{total}. Let further be

$$\mathbf{v} = (1.0, 0.5)$$

an infeasible solution in the velocity representation. The solution describes movement with full speed to the right for the first half of the total time, then movement with half of maximum speed to the right for the second half of the total time. Furthermore we assume the start position as absolutely left at 0.0. Going to the right with the speed of $1.0 \cdot v_{\max}$ will obviously result in the stacker being at the right end of the stockpile. In the second part the absolute position still increases with half ot the maximum speed until the absolute position value of 1.5 is reached. With

$$\lfloor 1.5 \rfloor \bmod 2 = 1.0$$

the mirrored position results in

$$p_{\text{mirrored}} = 1.0 - (1.5 - \lfloor 1.5 \rfloor) = 1.0 - (1.5 - 1.0) = 0.5$$

4.2.2 Position Representation

The positions representation does never exceed the path limits the same as velocity representation which never exceeds the speed limitations by design. But the speed limitations can be violated in the positions representation in the moment v_i in equation 4.5 exceeds the maximum speed $v_{\max}$ which can only happen if $\frac{|p_{i+1} - p_i|}{v_{\max}} > t^p_{\text{slot}}$. The possibility for invalid solutions increases the smaller t^p_{slot} gets. To tackle this issue two repairing mechanism are defined which utilize the maximum position difference $d_{\max} = t^p_{\text{slot}} \cdot v_{\max}$.

- The *direct correction* repairing iterates through the array once and limits each following position to the maximum difference to it's predecessor as shown in equation 4.7.

$$p_{i+1} = \begin{cases} p_i - d_{\max}, & \text{if } p_i - p_{i+1} > d_{\max}, \\ p_i + d_{\max}, & \text{if } p_{i+1} - p_i > d_{\max}, \\ p_{i+1}, & \text{otherwise.} \end{cases} \tag{4.7}$$

- *Iterative balancing* also iterates though the array but does not only change the successor p_{i+1} to each position p_i but also the current position half of the distance $d_{\text{corr}} = |p_i - p_{i+1}| - d_{\max}$ which has to be corrected. For each pair of consecutive positions the corrected positions are defined as described in equation 4.8.

$$(p_i, p_{i+1}) = \begin{cases} (p_i - \frac{d_{\text{corr}}}{2}, p_{i+1} + \frac{d_{\text{corr}}}{2}), & \text{if } p_i - p_{i+1} > d_{\max}, \\ (p_i + \frac{d_{\text{corr}}}{2}, p_{i+1} - \frac{d_{\text{corr}}}{2}), & \text{if } p_{i+1} - p_i > d_{\max}, \\ (p_i, p_{i+1}), & \text{otherwise.} \end{cases} \tag{4.8}$$

  The iteration converges eventually towards a feasible solution.

Example 4.2. Let the maximum distance between two positions be $d_{\max} = 0.4$ and a solution for the stacker traverse path in the positions representation $\mathbf{p} = (0.0, 0.4, 1.0)$. The solution consists of two movements between the three positions $p_1 = 0.0$, $p_2 = 0.4$ and $p_3 = 1.0$. As we perform the *iterative balancing* we compare p_1 and p_2 and get $p_1 - p_2 <= d_{\max}$ and $p_2 - p_1 <= d_{\max}$ - no repairing required. In the next step p_2 and p_3 are evaluated resulting in a distance $p_3 - p_2 = 0.6 > d_{\max}$.

To repair the individual the correction distance $d_{\text{corr}} = |0.6| - d_{\max} = 0.2$ is calculated and both positions are corrected half of the distance. This results in the partially corrected solution $(0.0, 0.5, 0.9)$. One can easily see that a new conflict between p_1 and p_2 arises in this individual which is the reason for this solution to be iterative.

Start		$(0.0, 0.4, 1.0)$
Iteration 1	$i = 1, \|p_1 - p_2\| \leq d_{\max}$	$(0.0, 0.4, 1.0)$
	$i = 2, d_{\text{corr}} = 0.2$	$(0.0, 0.5, 0.9)$
Iteration 2	$i = 1, d_{\text{corr}} = 0.1$	$(0.05, 0.45, 0.9)$
	$i = 2, d_{\text{corr}} = 0.05$	$(0.05, 0.475, 0.875)$
Iteration 3	$i = 1, d_{\text{corr}} = 0.025$	$(0.0625, 0.4625, 0.875)$
	$i = 2, d_{\text{corr}} = 0.0125$	$(0.0625, 0, 46875, 0, 86875)$

...

4.2.3 Repairing Application

An infeasible solution produced by the evolutionary algorithm repaired by the a mechanism can be handled in different ways:

- In the **Baldwinian** way the repaired solution is only used for the evaluation and not written back to the population individual. This means the population consists of feasible and infeasible individuals which are used equally for the evolution.

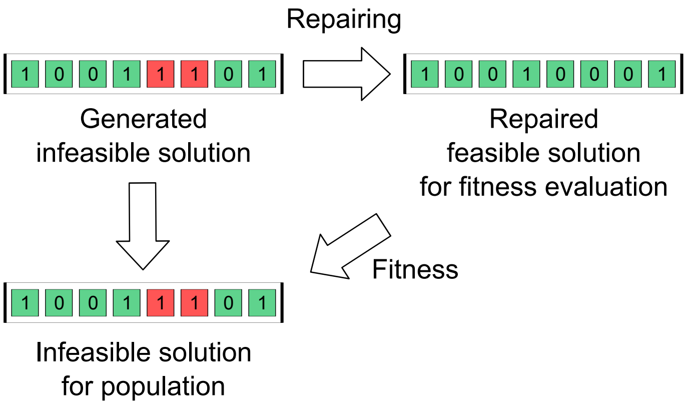


Figure 4.1: Illustration of Baldwinian repairing

- In the **Lamarckian** way the repaired solution is not only used for the representation but also written back to the population individual for further evolution. This means the population consists only of feasible individuals because each individual is repaired after creation.

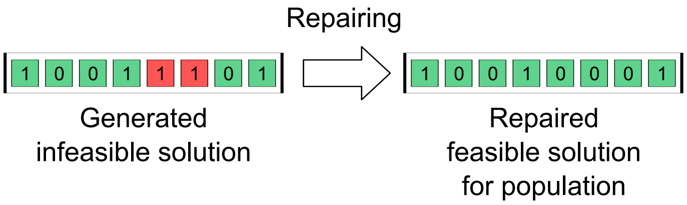


Figure 4.2: Illustration of Lamarckian repairing

In this thesis only the position representation is repaired using the Baldwinian and the Lamarckian way. If a solution in the velocity representation is infeasible and it is repaired the resulting individual is unlikely representable in the solution model as the mirroring takes place between two actual velocity changes. Lamarckian repairing would mean to change the amount of velocity changes or the fixed length of t^v_{slot} for each time slot and thus a huge difficulty for the model.

4.3 Fitness Evaluation

In order to evaluate a potential solution which does not violate the hard constraints fitness functions need to be defined to make multiple solutions comparable.

As the main optimization goal is the minimal variance of the quality output curve exactly this variance is defined to be one of the objectives for the optimization. Given

- the quality input curve $\mathbf{q}^{\text{in}}$,
- the valid stacker traverse path representation $\mathbf{p}'$,
- a freely definable number of cross sections n ablated by the reclaiming machine
- with each section having its specific amount of material w_i and its average quality q_i^{out},
- n' the amount of non-empty cross sections and
- $\overline{q}^{\text{out}}$ the total average quality

the fitness function F_1 is defined as described in equation 4.9.

$$F_1(\mathbf{q}^{\text{in}}, \mathbf{p}') := \frac{n' \sum_{i=1}^{n} w_i (q_i^{\text{out}} - \overline{q}^{\text{out}})^2}{(n'-1) \sum_{i=1}^{n} w_i}, \tag{4.9}$$

When the traverse path is changed it has an direct influence on the shape of the stockpile. This leads to the other objective to create a stockpile with a ridge of nearly constant height. The objective will be represented by the fitness function F_2 which indicates the relative height difference in the full width of the valid stacker traverse path as defined in equation 4.10 where $h_{\text{max}}, h_{\text{min}}$ and $\overline{h}$ denote the maximum, minimum, and the average stockpile heights respectively.

$$F_2(\mathbf{q}^{\text{in}}, \mathbf{p}') := \frac{h_{\text{max}} - h_{\text{min}}}{\overline{h}}, \tag{4.10}$$

The two objectives F_1 and F_2 combined define a standard multi-objective optimization problem [Deb01]. As typical in such kind of problems no single optimal solution can be calculated as usually there can be found a solution which outperforms the alleged best solution in one of the two objectives. The formulation as a multi-objective problem allows the derivation of insights about the trade-off necessary in one objective to get an improvement in another objective without having to specify this trade-off in advance as one would have to formulate it as a single-objective problem.

In order to do the multi-objective optimization and get insights from the set of partially optimal solutions the Pareto front [Deb01] is calculated which describes the set of non-dominated solutions found by the optimization in one run. A non-dominant solution is a solution which is not outperformed by another solution in all objectives.

It can be seen easily that the solutions form a balanced curve in the two dimensional objective space. This means that improving on one objective has a negative effect on the other objective. This results in a so-called knee region [DG11] which describes the solutions near to optimal in both objectives as it can be seen in figure 5.8. Moving out of this knee region a small gain in one objective has always to come with a high sacrifice in the other objective.

4.4 Result Set Comparison

By describing the result as explained in section 4.3 using the Pareto front found by an optimization system many solutions form the objective-balancing curve. This Pareto front does not have to consist of solutions absolutely optimal for their relation as there might be dominating ones which simply were not found by the optimization algorithm.

To evaluate the set of solutions as a whole and make it comparable to another set in order to evaluate the optimization algorithm two methods are used in this thesis:

- The **distance to non-dominated set** describes a measure in which a non-dominated set of solution is defined as a reference set. Each solution is measured against this set by first calculating the minimum distance to the reference set

for each point and using the root mean square of these values as a distance measure and thus as a single-number evaluation for the analyzed solution. The non-dominated set is problem-specific and thus can not be calculated once for all problems. Section 5.6 shows the non-dominated sets used for evaluating the results in this thesis.
Equation 4.11 describes the set distance with $\mathbf{r}$ being the non-dominated reference set, $\mathbf{f}$ the fitness values of the currently evaluated solution, $|\mathbf{r}|$, $|\mathbf{f}|$ the associated set sizes and $\mathrm{d}(\mathbf{x}, \mathbf{y})$ the Euclidean distance regarding the objective values as equally scaled.

$$\mathrm{d}_{\mathrm{set}}(\mathbf{r}, \mathbf{f}) = \sqrt{\frac{1}{|\mathbf{f}|} \cdot \sum_{f_i \in \mathbf{f}} (\min_{r_i \in \mathbf{r}} \mathrm{d}_{\mathrm{Euclid}}(f_i, r_i))^2} \tag{4.11}$$

Obviously a solution is better than another if the set distance is smaller.

- The **hypervolume** [Beu12] measure is used for comparison of solution sets where the problem is varied. For instance when the amount of variables or the maximum allowed speed is changed the set distance is not applicable the hypervolume measure can be used instead. The hypervolume utilizes a user-set reference point f^{ref} which puts on a weight on the different objectives. Then the volume of the space is calculated containing all dominated solutions by the solution set which themselves dominate the reference point. A solution is regarded as better the bigger the hypervolume is. An illustration of a hypervolume can be found in figure 4.3. Here the hypervolume is calculated by the area as the problem is bi-objective. The formula describing the calculation for the bi-objective solution space is shown in equation 4.12. For the calculation the solution set $\mathbf{f}$ has to be sorted descendingly by F_2 and thus ascendingly by F_1. Furthermore all solutions have to be removed from the set which exceed the reference point in at least one objective. $f_o^{(i)}$ describes the o-th objective of the i-th solution in the solution set $\mathbf{f}$ with $|\mathbf{f}|$ the amount of solutions in the prepared set.

$$\mathrm{HV}(\mathbf{f}, f^{\mathrm{ref}}) = (f_1^{\mathrm{ref}} - f_1^{(1)}) \cdot (f_2^{\mathrm{ref}} - f_2^{(1)}) + \sum_{i=2}^{|\mathbf{f}|} (f_1^{\mathrm{ref}} - f_1^{(i)}) \cdot (f_2^{(i-1)} - f_2^{(i)}) \tag{4.12}$$

4.5 Evolutionary Optimization

As the complex nature of the optimization problem optimizing the stacking traverse path regarding the objectives defined in section 3.3 can not be described in a closed mathematical form a black box optimization system is required to solve the optimization problem. In the last two decades evolutionary optimization algorithms have evolved to adequate methods solving problems where no discrete algorithm can be expressed as they provide a flexible optimization environment inspired by natural evolution [Deb01].

In detail the evolutionary algorithm used in this thesis is the steady-state version of the NSGA-II algorithm [DPAM02, DNLA09] which is illustrated in figure 4.4. The algorithm works iteratively with the following steps:

- **Step 0**: Generate a random population of individuals (solutions) P_0 of size N, ranking solutions based on Pareto dominance.

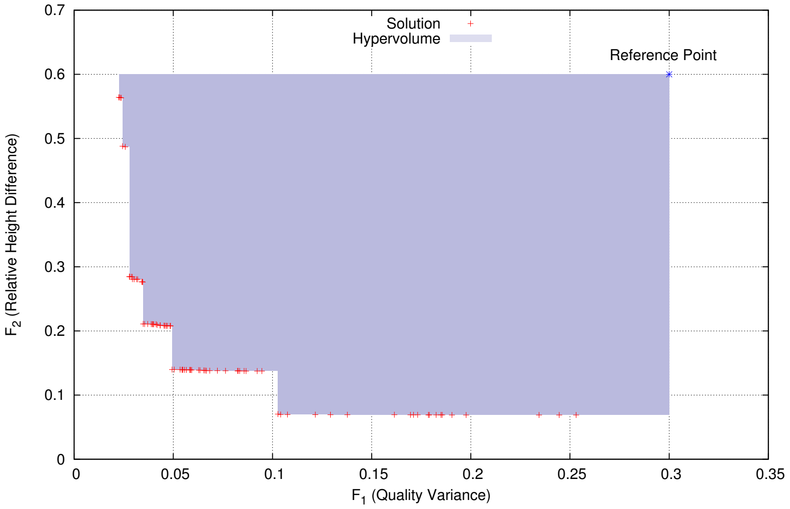


Figure 4.3: Illustration of a hypervolume

- **Step 1**: Generate one child q_t based on binary tournament selection, recombination and mutation.
- **Step 2**: Insert child into population P_t and recalculate the non-dominated sorting, identifying the different fronts.
- **Step 3**: In the last front perform crowding sorting [Deb01] and remove the solution one with the worst crowding distance value.
- **Step 4**: If the number of maximum evaluations is not reached go to **Step 1**.

Within this algorithm the operators and detailed parameters used are described in section 5.1.

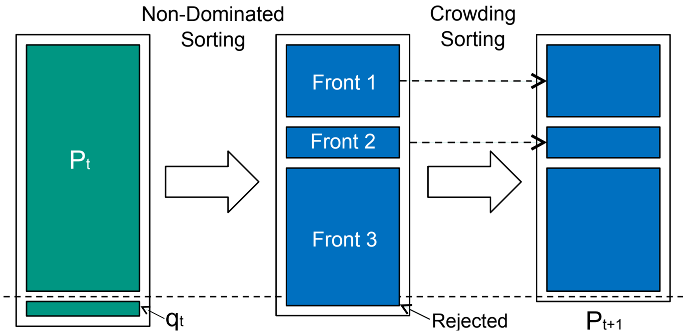


Figure 4.4: Illustration of the steady state version of the NSGA-II algorithm

Based on the knowledge of having a knee region of solutions which are of the highest value for the use of the optimization one could also use a specialized knee-finding algorithm as described by Shukla et al. [SBS13]. As the topic of this thesis is not only optimizing for the objectives but also getting insights about the solutions the NSGA-II algorithm is more suitable at the moment of time.

As the implementation framework the jMetal framework [jme13] was chosen.

4.5.1 Constraint Measure

Using the evolutionary optimization the repairing can also be completely abandoned and only a constraint measure can be defined. The measure puts a number on how far away a solution is from being feasible, the degree of constraint violation. This way the solution candidates can be sorted from feasible to infeasible to most infeasible. Feasible individuals are always preferred over infeasible individuals. Less infeasible individuals are preferred over more infeasible ones. The constraint measure is calculated in this thesis by summing up the distance between each original position p_i and the repaired position r_i as shown in equation 4.13.

$$c = \sum_i \text{abs}(p_i - r_i) \tag{4.13}$$

The repairing method used to create the corrected solution can be chosen freely. There is no writing of the repaired array, neither for the evaluation nor for the population.

The described constrained violation measure can not be used for the velocity representation as it requires a repaired solution in individual representation which is not possible with the repairing mechanisms defined in section 4.2. Thus another constraint measure has to be defined but is not part of this thesis. For the positions representation the results achieved with the constraint violation measure are are presented and evaluated in chapter 5.

Example 4.3. Let a solution in position representation calculated by the optimization system be

$$\mathbf{p} = (0.0, 0.4, 1.0, 1.0, 0.0)$$

and the maximum distance between two positions for a feasible solution be $d_{\max} = 0.4$. After repairing the individual for the constraint violation calculation with for instance the direct repairing mechanisms the solution results in being

$$\mathbf{r} = (0.0, 0.4, 0.8, 1.0, 0.6)$$

The constraint violation calculates from the sum of the absolute differences of the unrepaired and the repaired solution. In this case the constraint violation is

$$c = 0.0 + 0.0 + 0.2 + 0.0 + 0.6 = 0.8$$

4.6 Summary

In this chapter the algorithms used for optimizing the solutions for the described problem are presented. Within the algorithms different solution representations are used which have different advantages and disadvantages. The representations and the combinations of algorithms with repair mechanisms and constraint measures are applied combined to calculate the results presented in section 5.

5. Evaluation

This chapter describes the results achieved with the methods presented in section 4. As indicated by the goals of this thesis the optimization was researched in two directions:

- present a flexible system suitable for bulk material blending optimization and
- gain insights from solutions found by the optimization system.

As stated in section 3.7.2 one simulation run takes around 15 ms on a modern home computer to calculate. In total roughly 6,500 optimization runs were calculated for the generation of the data for this thesis. Each of these optimization runs was done with 25,000 individual simulation evaluations so in total approximately 162,500,000 simulations were executed which sums up to a total calculation time 677 hours if run on one modern CPU core with 3.8 GHz clock speed. To make the calculation of the many evaluation runs possible a self-implemented cluster computation software was used to utilize all available computers and increase the calculation speed.

Furthermore the calculations resulted in many single data files for which a special analysis software was implemented which generated most of the tables and visualizations used in this thesis.

The heap data used in this thesis is also taken from the real world coal processing plant which was mentioned in section 3.6. The stockpile dimensions are typical but not constant in this plant and were therefore used approximately.

Furthermore the typical coal angle of repose of 45 degrees [SPR11] was used in this thesis to simulate the stockpiles.

The following sections do not only present the optimization results outperforming the static blending techniques but also communicate a greater understanding for the way the algorithms work together with the optimization system.

5.1 Default Parameter Settings

Additionally to the operators described in section 4.5 table 5.1 shows the default parameters used for the evaluations if nothing else was stated in the specific evaluation section.

Parameter	**Value**
Population Size	100
Maximum Evaluations	25000
Mutation Probability	0.025
Crossover Probability	0.9
Heap Length	300 m
Heap Width	50 m
Maximum Speed	20 Layers
Variable Count	30
Quality Input Curve	$\mathbf{q}_1$
Reference Point	$F_1 = 0.3$, $F_2 = 0.6$

Table 5.1: Default parameters

Maximum evaluations describes the amount of iterations the NSGA-II algorithm has to create offspring including the generation of the initial population. *Mutation probability* represents the chance of one variable in a solution to be mutated. *Crossover probability* describes the overall chance of two solutions being recombinated. *Maximum speed* is given in layers as this states the speed comparable to a static stacking system. This means: if a static stacking system was moving all the time with maximum speed back and forth it could stack up to 20 layers until the stockpile was finished. *Variable count* describes the degree of freedom for the optimization system. For the velocity representation the variable count of 30 results in 30 different movement velocities in equally distributed time slots. For the velocity representation this describes 29 movements. *Quality input curve* chosen for illustration of certain examples is $\mathbf{q}_1$ if not stated differently. *Reference point* for the hypervolume calculation is set to $F_1 = 0.3$, $F_2 = 0.6$.

The operators used used are the standard real parameter SBX crossover operator and the polynomial mutation operator with $\eta_c = 15$ and $\eta_m = 20$ respectively.

5.2 Quality Data

The quality input curves $\mathbf{q}_1 \ldots \mathbf{q}_{11}$ used for the evaluations are shown in figure 5.1 on page 29. The graphs also contain the information about the mean value and the quality variance for the input curve represented in form of the standard deviation.

The curves differ in standard variation, amount and distribution of nearly stable levels and mean value and are the base for all the investigation presented in this thesis.

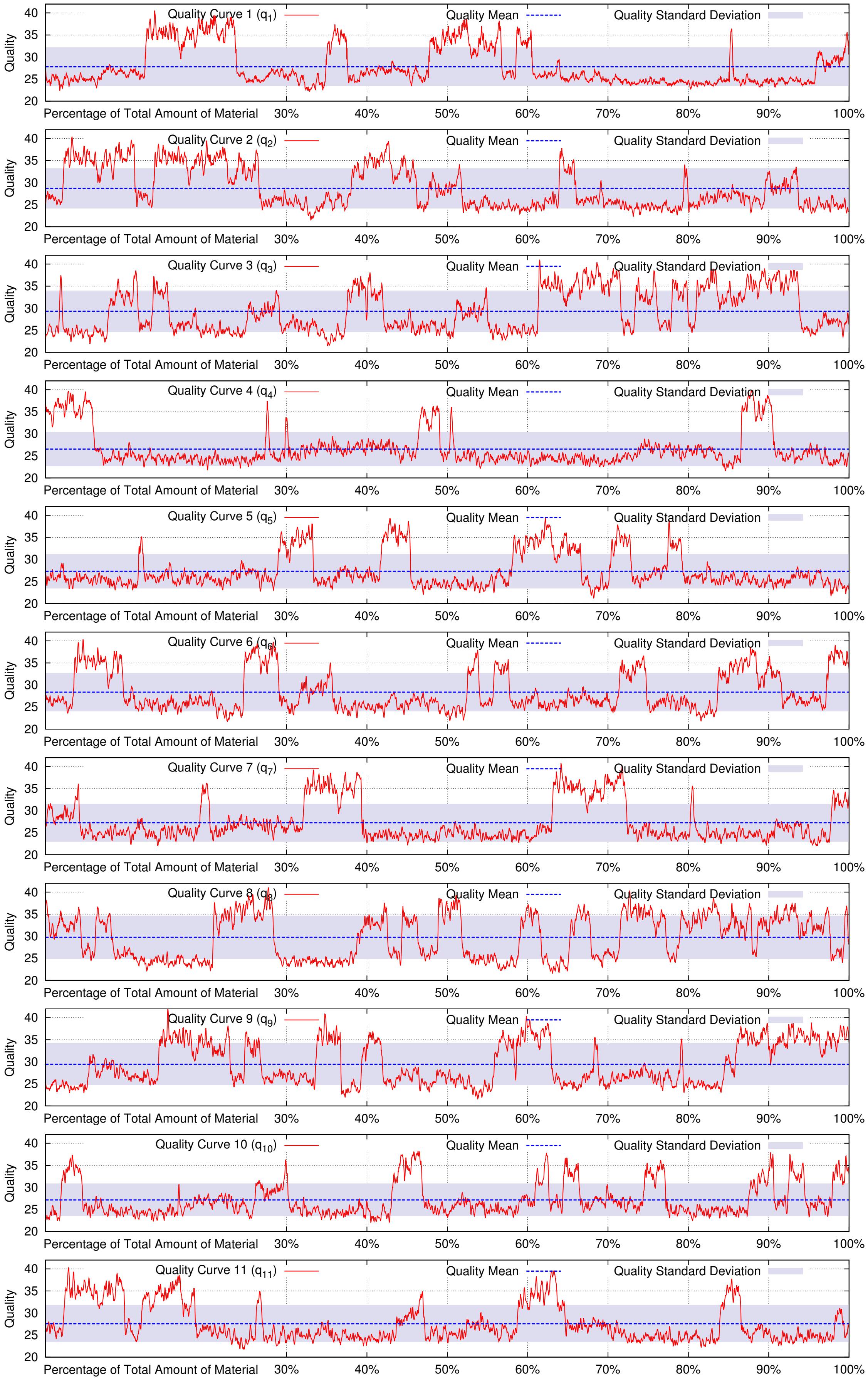


Figure 5.1: All quality input curves used in the following sections

5.3 Algorithm Naming

The representation models presented in section 4.1.1 and 4.1.2 combined with the repair mechanisms described in section 4.2 repairing either in the Lamarckian or the Baldwinian way as shown in section 4.2.3 combined make a large sum of final algorithms. In table 5.2 the implemented and evaluated algorithms are composed to give an overview of the abbreviations used in the following sections. The descriptive short indicators for the algorithm differentiation are explained in detail in the referenced sections.

Algorithm	**Representation**	**Repairing**	
		Method	**Application**
$\mathcal{A}^{v}_{\text{Mirr,Bal}}$	Velocity	Mirroring	Baldwinian
$\mathcal{A}^{p}_{\text{Dir,Bal}}$	Position	Direct	Baldwinian
$\mathcal{A}^{p}_{\text{Iter,Bal}}$	Position	Iterative	Baldwinian
$\mathcal{A}^{p}_{\text{Dir,Lam}}$	Position	Direct	Lamarckian
$\mathcal{A}^{p}_{\text{Iter,Lam}}$	Position	Iterative	Lamarckian
$\mathcal{A}^{p}_{\text{Dir,Mix}}$	Position	Direct	Baldwinian & Lamarckian
$\mathcal{A}^{p}_{\text{Iter,Mix}}$	Position	Iterative	Baldwinian & Lamarckian
$\mathcal{A}^{p}_{\text{Dir,Con}}$	Position	Direct	Constraint Violation
$\mathcal{A}^{p}_{\text{Iter,Con}}$	Position	Iterative	Constraint Violation

Table 5.2: Mapping of the algorithm names

5.4 Exemplary Results

This section give a quick overview over the different types of intermediate and final results generated by the optimization system to communicate an understanding for the data presented in the following sections.

The main input of the parametrized optimization system is the quality input curve. An exemplary curve is shown once again in figure 5.2 for comparison with the results. Levels of nearly constant quality can be seen in the curve. These levels need to be equalized as the fast low amplitude variation of the quality is not of such a big interest as the low frequency variation which creates these levels. Therefore F_1 (quality variance) was defined which is a measure for the average distance from the mean quality value. Mean quality value and standard deviation are also shown in figure 5.2.

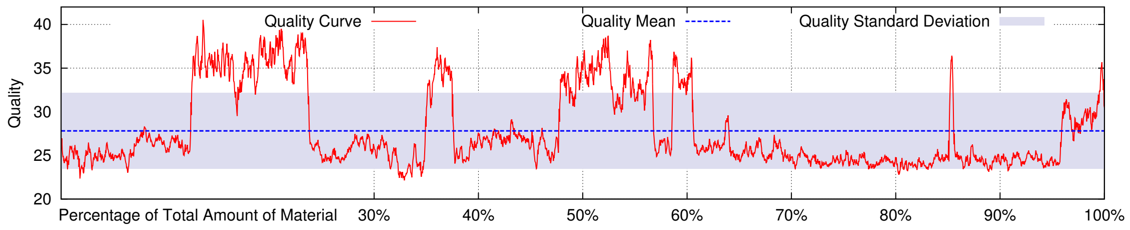


Figure 5.2: Exemplary quality curve $\mathbf{q}_1$

The optimization system generates solutions based on the algorithms described in section 4.5. These solutions are then evaluated for their fitness values. This evaluation is done in the fast simulation system described in section 3.7.2. The output of this simulation system is the quality output curve but also the stockpile height shape for calculation of the second objective, the relative height difference.

Figure 5.3 shows an exemplary stockpile height shape as well as the corresponding traverse path which was used building this heap. The traverse path looks quite chaotic on first sight because it is out of context of the quality input curve.

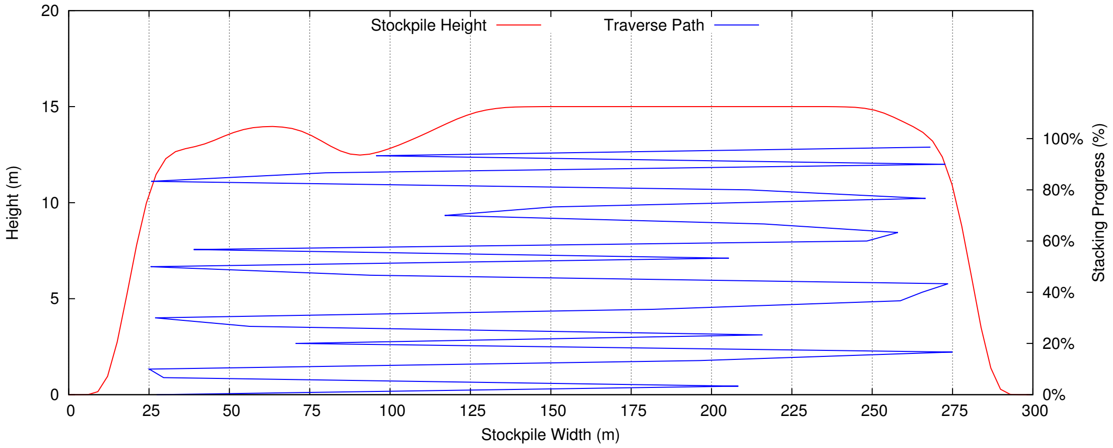


Figure 5.3: Exemplary stockpile with illustrated stacker traverse path

In figure 5.4 the traverse path is drawn vertically together with the quality input curve. This way it is easy to understand the solution and why it is optimizing the minimum variance. The first two high quality levels are distributed mainly on the right side of the stockpile. When the next high quality level comes to be distributed on the stockpile the stacker mainly drives on the left side. Plotting the data this way a user can easily follow the optimized solution.

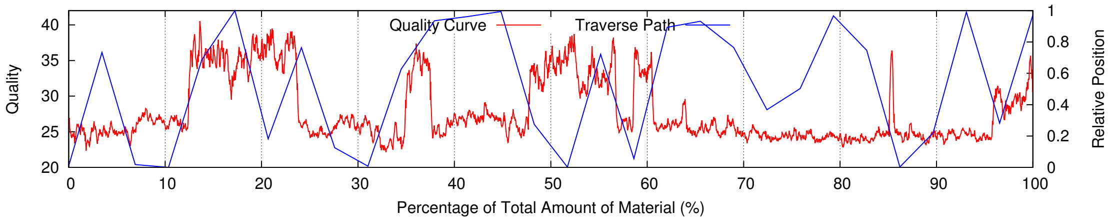


Figure 5.4: Exemplary traverse path illustrated together with the corresponding quality input curve

Up to now only one solution was shown. The optimization system is sorting the solutions in the population to Pareto front, second, third and other fronts and generates child generations from the population. This development of solutions is shown in figures 5.5 and 5.6. There the generated solutions are evaluated for their fitness values and the fitness values are plotted on the different y axes. For orientation the default reference point values were inserted. The steps of the F_2 values are clearly visible which come from the integer height discretization in the fast simulation. The

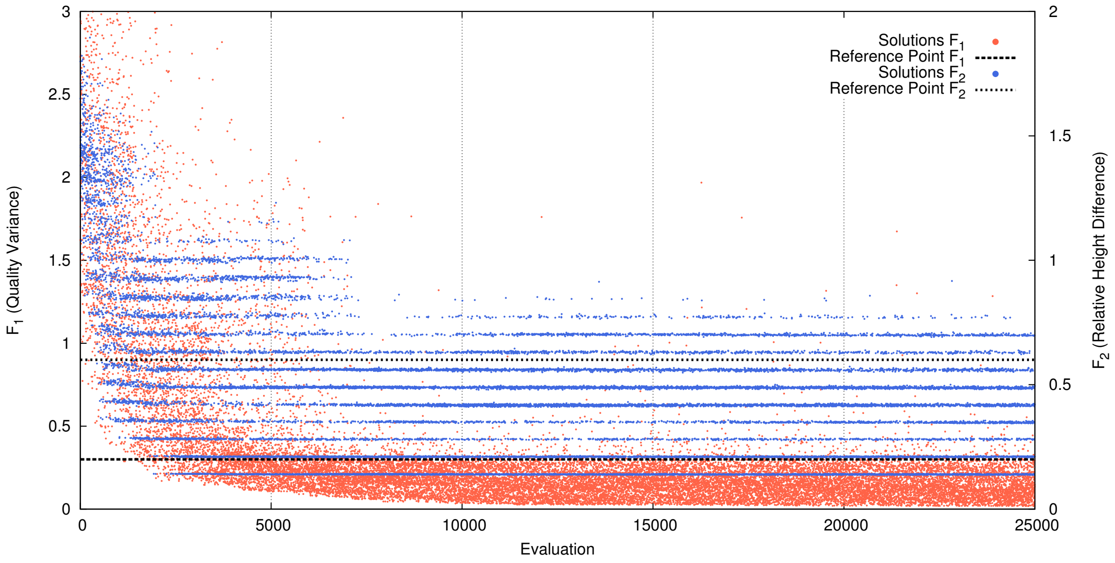


Figure 5.5: Exemplary child generation development for both objectives calculated with $\mathcal{A}^{p}_{\text{Dir,Bal}}$

convergence of the algorithm towards good solutions especially regarding the F_1 values is also visible.

Same algorithm but a different run also produced the results shown in the second figure which looks much more chaotic and random at first sight. Bit still a permanent positive development of the best new solutions generated can be observed which is the key difference to purely random search.

As the optimization system has two objectives the final result is always a solution set consisting of the Pareto front found by the current optimization run. Exemplary Pareto fronts found in different runs can be found in figure 5.7.

Again the step like structure in the objective graph can be observed. F_2 describes the relative height difference. As the fast simulation is calculating on integer values for x, y and z positions of the particles the resulting height of the stockpile is also an integer value. Only the average height is a floating point value which explains why there are multiple relative height difference values on one integer level of absolute height difference.

After evaluation of many solution sets the formation of a knee region with both objectives nearly optimal but still in balance of the objectives can be seen in figure 5.8. This knee region is of special interest if one is only optimizing the system. But as one of this thesis' goals is the research of the partially optimal solutions the whole Pareto front is of interest.

5.5 Comparison to Static Stacking

The solutions found by the optimization system always outperformed the solutions generated by static stacking methods. Figure 5.9 presents the output quality curve generated with static stacking with 20 layers of material. Even though this is driving with maximum speed all the time back and forth compared to the optimized solution

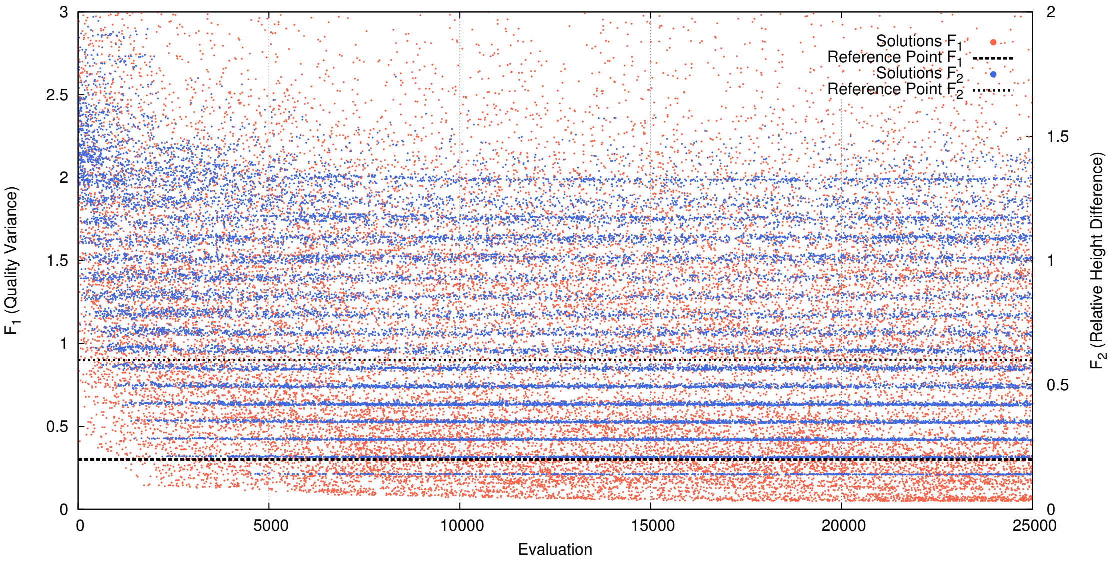


Figure 5.6: Second exemplary child generation development for both objectives calculated with $\mathcal{A}^{p}_{\text{Dir,Bal}}$

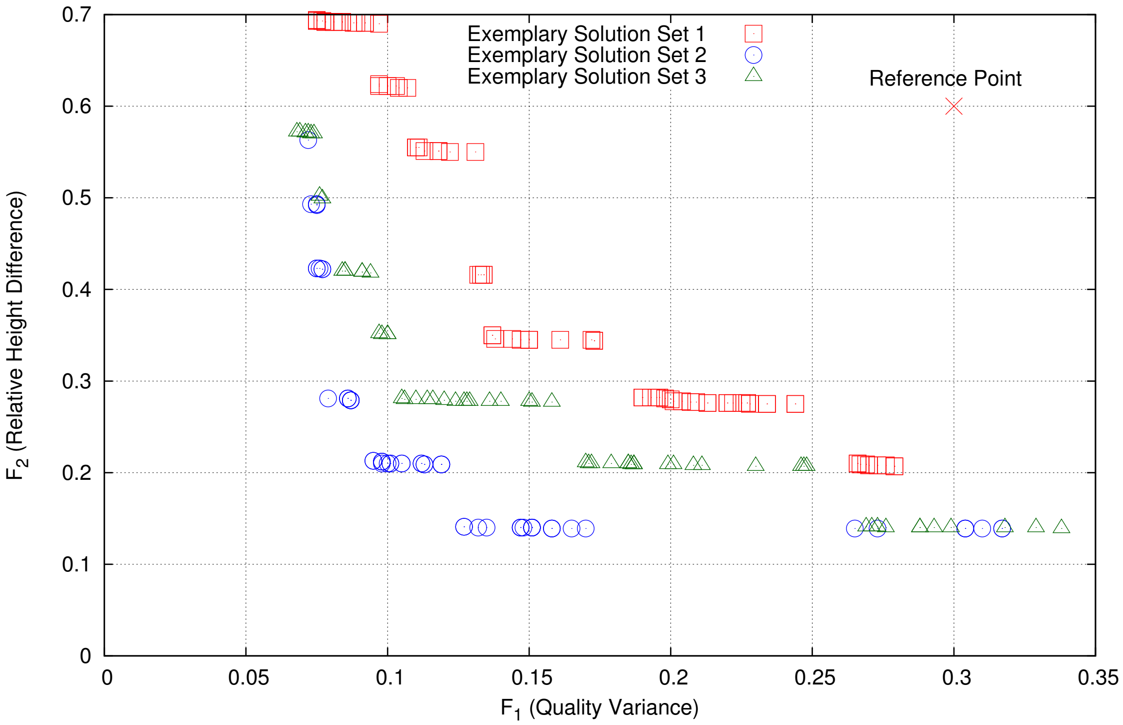


Figure 5.7: Exemplary Pareto fronts found in different executions of the optimization system

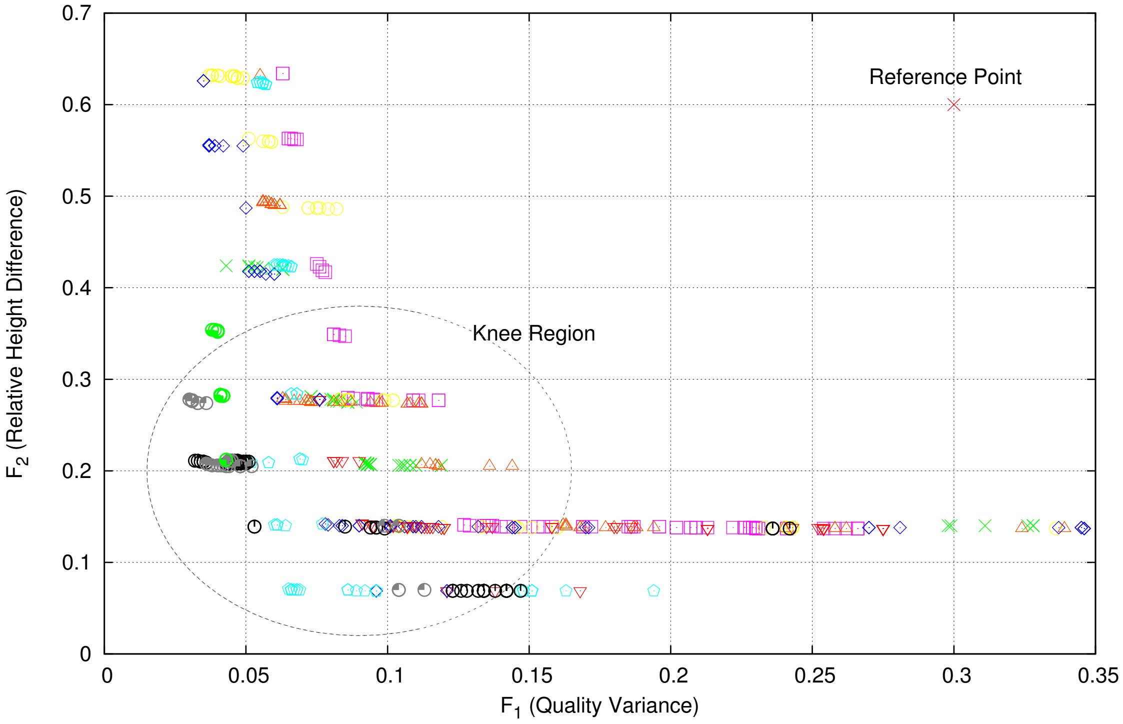


Figure 5.8: Exemplary solutions sets illustrating the knee region balancing the two objectives

the output quality curve does not look nearly optimal as there are still levels and wider peaks of qualities far from the mean value.

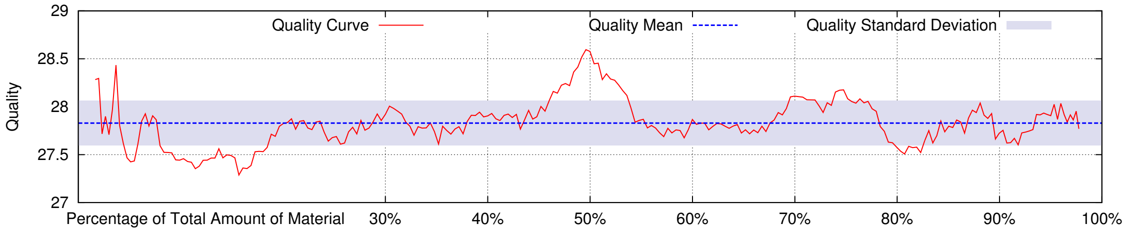


Figure 5.9: Output quality curve generated with static stacking with 20 layers of material

Figure 5.10 shows an exemplary quality output curve from the knee region. This curve has different statistical properties. A higher frequency variation (which is not of interest for the process) can still be observed but no extended levels of qualities far from the mean value exist in the curve. Peaks are thin and come back to the mean value within short time.

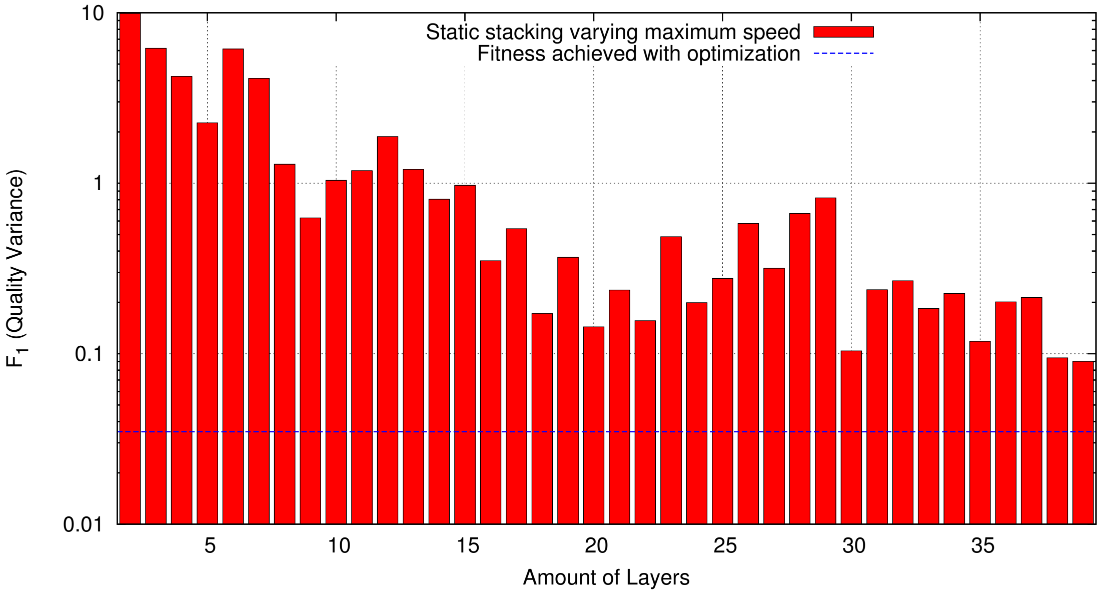


Figure 5.11: Static stacking with different speeds

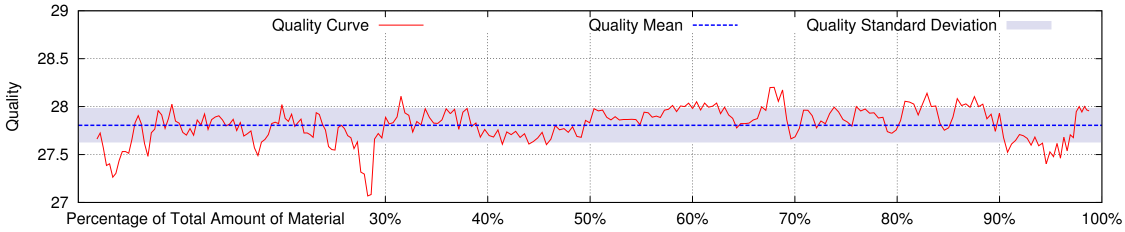


Figure 5.10: Exemplary optimized quality output curve

To further illustrate advantage of the optimized solutions over the ones generated with static stacking figure 5.11 illustrates the development of the fitness value for F_1 for the static stacking compared to a random solution from the knee region found by the optimization algorithm.

The figure which is drawn in the log scale illustrates clearly the disadvantages of the static stacking. Not only does the static stacking never reach a fitness value close to the value calculated for the optimized solution but the solutions get even worse after increasing the speed to more than 20 layers. Only after further increasing the speed to 30 layers and more (note here the optimized solution is working with a maximum speed of 20 layers) the solutions get better fitness values again.

To further show how the optimization outperforms static stacking table 5.3 presents the F_2 values of knee solutions which were found during the optimization runs. The simple fact that each optimization run finds solutions outperforming static stacking is already a proof that the system is suitable for producing good blending results. The entry values in the table indicate the fitness value for the relative height difference. There a high value means a high variation in the stockpile height. As this objective is not limited as a hard constraint even the unstable algorithms $\mathcal{A}^p_{\text{Iter,Lam}}$, $\mathcal{A}^p_{\text{Dir,Con}}$and $\mathcal{A}^p_{\text{Iter,Con}}$always find solutions which are outperforming the static stacking but get bad results in terms of height difference. The high stability of the algorithms $\mathcal{A}^p_{\text{Dir,Bal}}$and $\mathcal{A}^p_{\text{Iter,Bal}}$is clearly visible which make them the most valuable algorithms.

		Algorithms						
		$\mathcal{A}^v_{\text{Mirr,Bal}}$	$\mathcal{A}^p_{\text{Dir,Bal}}$	$\mathcal{A}^p_{\text{Iter,Bal}}$	$\mathcal{A}^p_{\text{Dir,Lam}}$	$\mathcal{A}^p_{\text{Iter,Lam}}$	$\mathcal{A}^p_{\text{Dir,Con}}$	$\mathcal{A}^p_{\text{Iter,Con}}$
Runs	**1**	0.279	0.277	0.211	0.211	0.209	0.624	0.208
	2	0.277	0.417	0.139	0.208	0.704	0.482	0.208
	3	0.353	0.355	0.421	0.207	0.346	0.276	0.141
	4	0.562	0.209	0.491	0.206	0.279	0.481	0.277
	5	0.344	0.206	0.140	0.277	0.208	0.349	0.705
	6	0.341	0.138	0.207	0.140	0.482	0.347	0.554
	7	0.280	0.278	0.211	0.207	0.720	0.208	0.346
	8	0.563	0.207	0.207	0.412	0.345	0.704	0.279
	9	0.275	0.140	0.139	0.553	0.281	0.282	0.346
	10	0.281	0.280	0.346	0.422	0.351	0.699	0.620

Table 5.3: F_2 value for knee solutions outperforming static stacking

5.6 Non-Dominated Set for Distance Measure

In section 4.4 the need for a set measure was described. For comparison of different problems the hypervolume is used but to have a more meaningful measure within a specific problem the non-dominated sets were calculated to allow the calculation of a set distance to a true Pareto front for the results within the problem.

As a true Pareto front can not be described explicitly for the problem the non-dominated set was calculated over all solutions in all solutions sets ever run with the optimization system. The calculation was done by unifying all solutions and calculating the Pareto front over all the solutions. It is to note at this point that each solution was already part of a Pareto front found by the optimization system after evaluating 25000 individuals.

To calculate the non-dominated set for the quality input curve $\mathbf{q}_1$ roughly $112,000,000$ [$\approx 4,474$ Runs$\cdot 25,000$ Evaulations] individuals were considered of which figure 5.12 contains only the ones which were included in at least one final Pareto front.

The figure also includes the non-dominated set and piecewise linear interpolations for the set. The interpolations are necessary to create a meaningful measure for the distance of a solution set to the non-dominated set. For example there are no solutions in the non-dominated set around the value $F_2 = 1$ but for a newly generated solution set it would not make sense to calculate the distance to the true solution below or above this position. Therefore the interpolations describe additional solutions in the non-dominated set at positions where in some cases actual solutions could not even have a corresponding value (cf. the interpolated solutions around $F_1 = 0.6$).

It is highly interesting to see of which solutions the non-dominated set consists. This allows to understand the problem in greater detail and gives insights about the construction of Pareto optimal solutions. Furthermore direct parameter combinations can extracted which are responsible for generating the best solution sets.

The following facts could be extracted from the solutions:

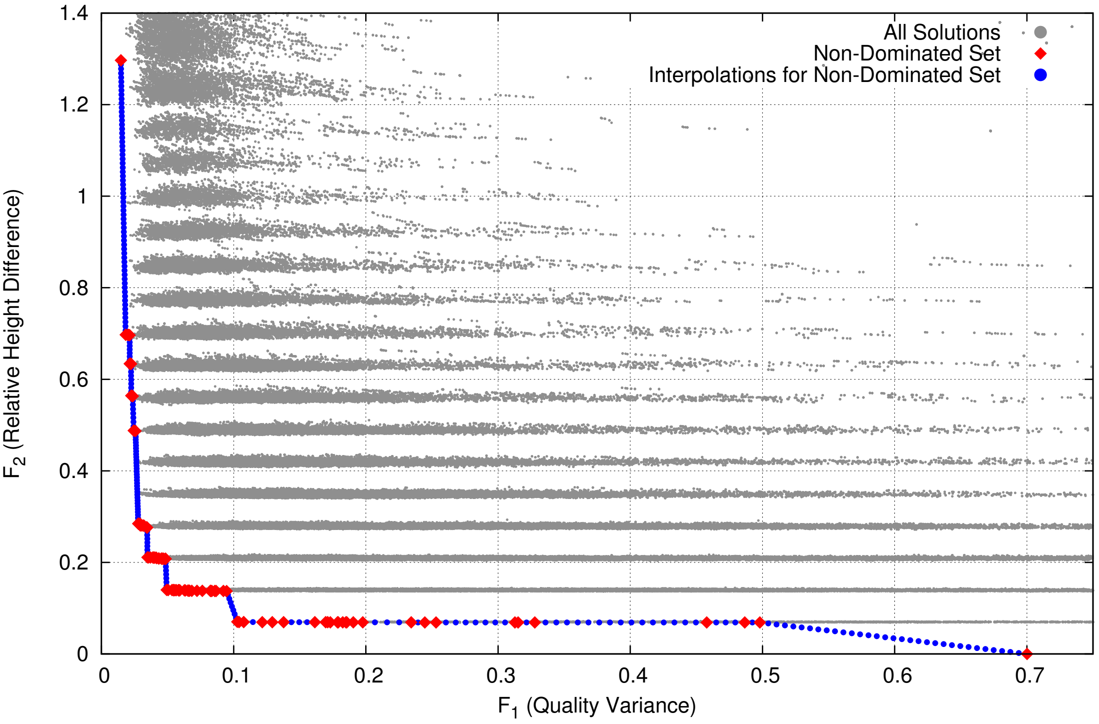


Figure 5.12: Illustration of the non-dominated set for quality input curve $\mathbf{q}_1$

- 96% of the solutions in the non-dominated set were calculated during the evaluation of different mutation and crossover probabilities whereas these calculations only form 87% of the total calculations done for quality curve $\mathbf{q}_1$.
- Of the solutions in the non-dominated set calculated during the probability assessment 80% were calculated using a crossover probability of 0.8 or 1.0 and mainly form the knee area. The 20% left were calculated with lower crossover probabilities but on average with higher mutation probability and form the extremes of the non-dominated set.
- The solution with the best F_1 value in the non-dominated set was calculated during the population variation calculations with a population size of 10 individuals using algorithm $\mathcal{A}^p_{\text{Dir,Bal}}$.
- The remaining 4% of solutions in the non-dominated set were calculated during runs with default parameters using algorithm $\mathcal{A}^p_{\text{Dir,Bal}}$.
- The solutions in the non-dominated set were calculated to 41% with $\mathcal{A}^p_{\text{Dir,Bal}}$, 29% with $\mathcal{A}^p_{\text{Dir,Lam}}$, 12% with $\mathcal{A}^p_{\text{Iter,Bal}}$, 8% with $\mathcal{A}^p_{\text{Iter,Lam}}$and 10% with $\mathcal{A}^v_{\text{Mirr,Bal}}$as visualized in figure 5.13.

These results present insights about the value of the different algorithms. Obviously the most important algorithms for finding the most valuable results for the optimization are the ones with direct repairing method and of these the one with Baldwinian repairing. This is proven also shown in more detail in section 5.7.

The next interesting topic to analyze is the construction of the solutions in the non-dominated set. To analyze the data in groups the solutions in the set were partitioned

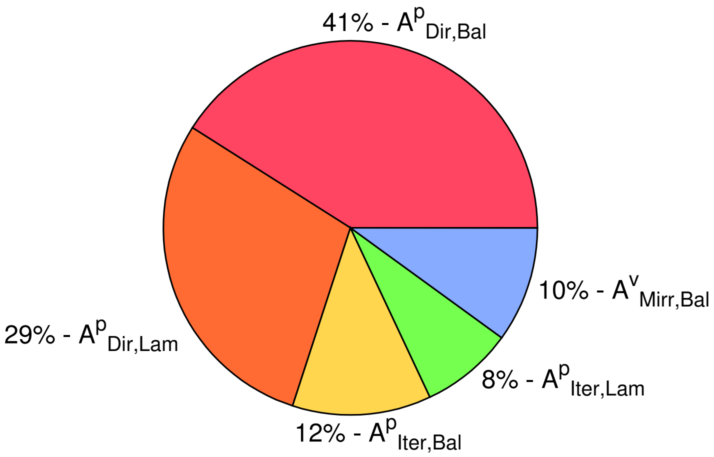


Figure 5.13: Illustration of the algorithm fragmentation of the non-dominated set

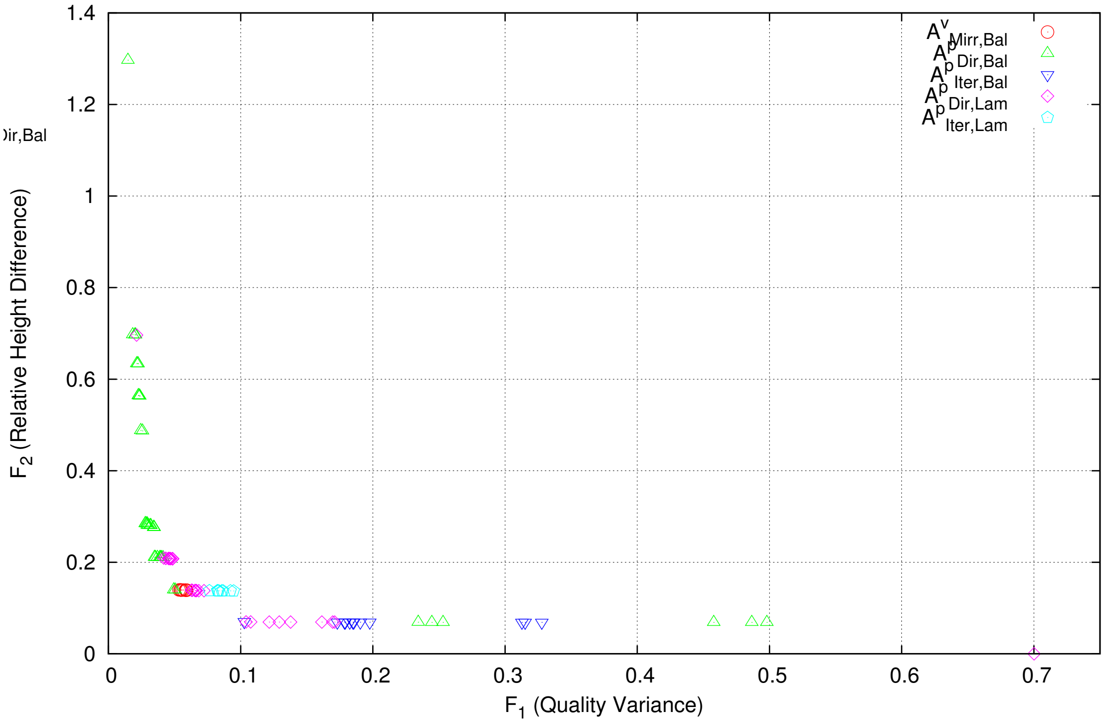


Figure 5.14: Illustration of the algorithm fragmentation of the non-dominated set in objective graph

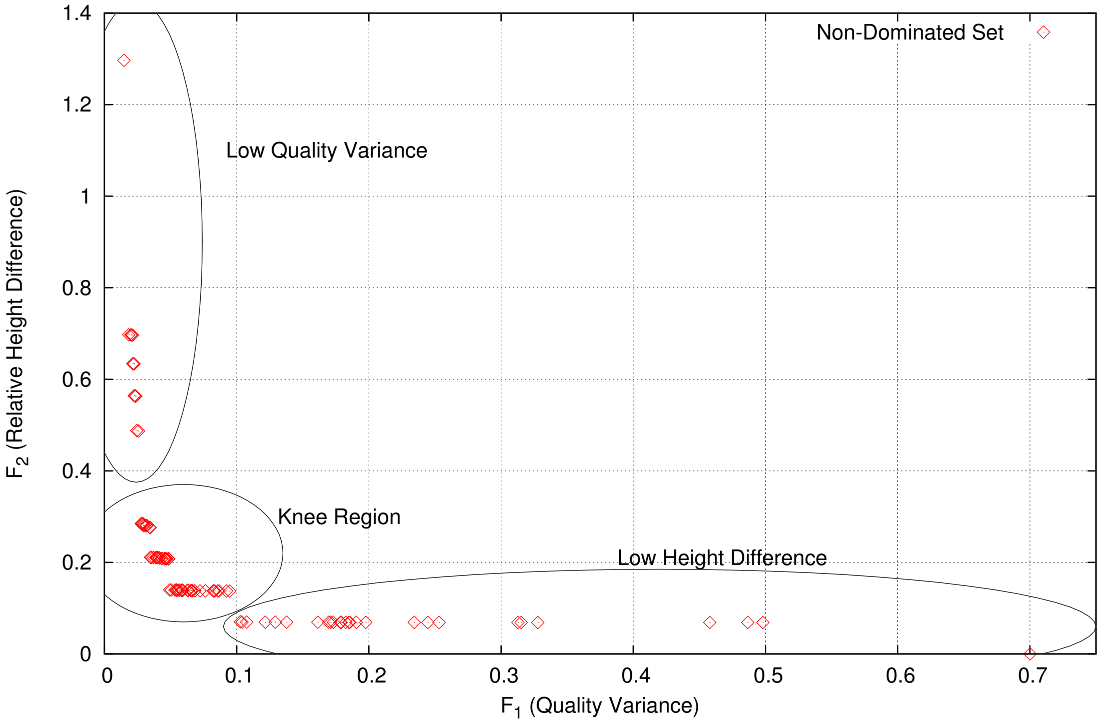


Figure 5.15: The partitions of the non-dominated set

to results with low quality variance, results in the knee region and results with low height difference as illustrated in figure 5.15.

Within these groups histograms for position distribution and and speed distribution were calculated and are shown in figure 5.16 and 5.17.

The position distribution displays as expected: the positions at the borders are highly frequented by all of the solutions as the algorithm has to fill up the stockpile till the borders to get a low height difference. All other positions are visited on the way but the border positions have to be accessed explicitly that's why they are noted with the highest percentage.

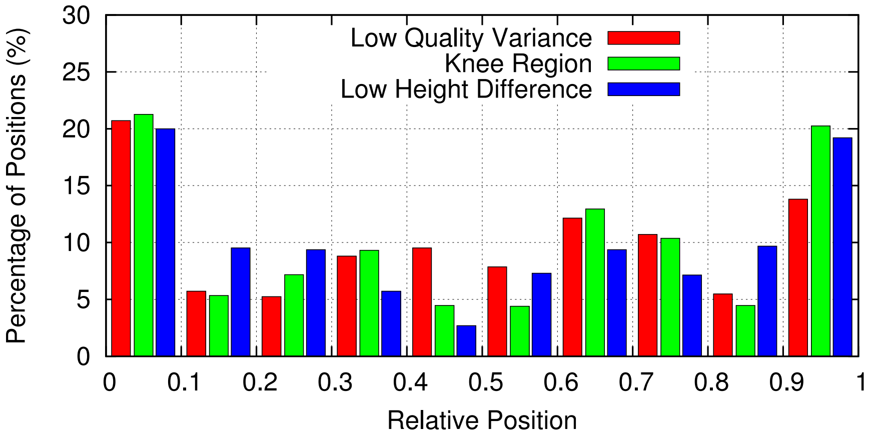


Figure 5.16: Position distribution of the solutions in the non-dominated set

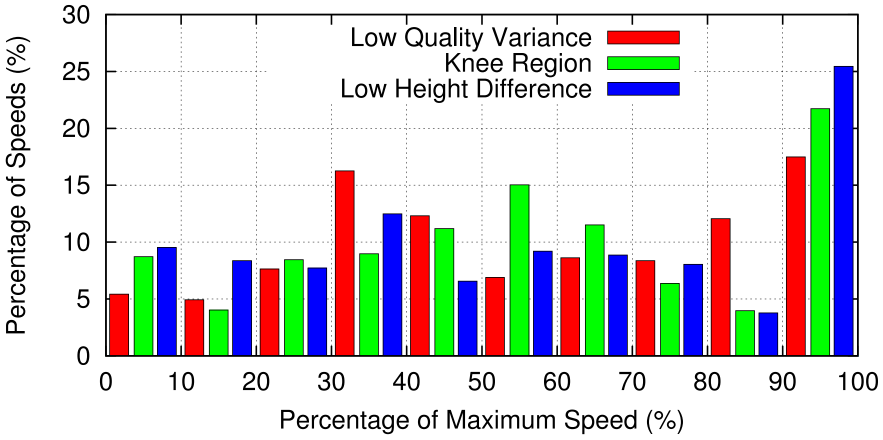


Figure 5.17: Speed distribution of the solutions in the non-dominated set

The speed distribution shows a behavior which needs further explanation. The solutions with low quality variance use a lower average speed and less maximum speed than all other solutions. The solutions optimizing the height difference show the highest speeds. One explanation for this observation is the fact that the higher maximum speed usage is closer to the static stacking and thus to the perfect shape with the lowest height difference. The objective of minimizing the quality variance though explains why there is not always maximum speed driving as sometimes it is required to get to an intermediate position to optimize the material quality spread.

5.7 Algorithm Observations

In the previous section already some insights could be found about the solutions and the algorithms. Table 5.4 shows the median hypervolumes achieved with the different algorithms for different quality curves. The table clearly shows the algorithms with the best results as $\mathcal{A}^p_{\text{Dir,Bal}}$, $\mathcal{A}^p_{\text{Iter,Bal}}$and $\mathcal{A}^p_{\text{Dir,Lam}}$. The algorithms $\mathcal{A}^p_{\text{Iter,Lam}}$and $\mathcal{A}^p_{\text{Iter,Con}}$still see to produce acceptable results. $\mathcal{A}^v_{\text{Mirr,Bal}}$and $\mathcal{A}^p_{\text{Dir,Con}}$show the worst results with regard to the median hypervolume.

In the table 5.5 though the standard deviations of the hypervolumes for different independent runs are displayed. Here the performance is slightly different. Besides being one of the algorithms with the worst results $\mathcal{A}^v_{\text{Mirr,Bal}}$performs very reliably. The suitability of $\mathcal{A}^p_{\text{Dir,Bal}}$and $\mathcal{A}^p_{\text{Iter,Bal}}$is confirmed but the biggest difference is given for $\mathcal{A}^p_{\text{Iter,Con}}$where the results were acceptable but the second table shows a bad stability of the results. $\mathcal{A}^p_{\text{Dir,Con}}$is confirmed as not suitable.

5.8 Parameter Variation

To gain a deeper understanding for the behaviour of the algorithms and their properties different parameters were varied and are presented in the following subsections.

5.8.1 Population Size

The population size is the amount of the co-existing individuals (solutions) in one optimization run. Each child is generated from this population and then mixed from

		Algorithms						
		$\mathcal{A}^{v}_{\text{Mirr,Bal}}$	$\mathcal{A}^{p}_{\text{Dir,Bal}}$	$\mathcal{A}^{p}_{\text{Iter,Bal}}$	$\mathcal{A}^{p}_{\text{Dir,Lam}}$	$\mathcal{A}^{p}_{\text{Iter,Lam}}$	$\mathcal{A}^{p}_{\text{Dir,Con}}$	$\mathcal{A}^{p}_{\text{Iter,Con}}$
Quality Input Curves	$\mathbf{q}_1$	0.062	0.083	0.090	0.084	0.067	0.062	0.078
	$\mathbf{q}_2$	0.056	0.077	0.077	0.048	0.059	0.037	0.072
	$\mathbf{q}_3$	0.033	0.073	0.061	0.068	0.063	0.032	0.056
	$\mathbf{q}_4$	0.079	0.111	0.098	0.090	0.075	0.083	0.084
	$\mathbf{q}_5$	0.072	0.084	0.102	0.103	0.069	0.070	0.080
	$\mathbf{q}_6$	0.041	0.079	0.054	0.075	0.051	0.019	0.058
	$\mathbf{q}_7$	0.067	0.091	0.076	0.086	0.064	0.050	0.071
	$\mathbf{q}_8$	0.034	0.061	0.053	0.068	0.052	0.060	0.048
	$\mathbf{q}_9$	0.043	0.046	0.046	0.056	0.062	0.039	0.039
	$\mathbf{q}_{10}$	0.069	0.089	0.096	0.090	0.079	0.080	0.086
	$\mathbf{q}_{11}$	0.056	0.086	0.068	0.078	0.066	0.059	0.067

Table 5.4: Hypervolumes of median solutions achieved for different input quality curves with different algorithms

		Algorithms						
		$\mathcal{A}^{v}_{\text{Mirr,Bal}}$	$\mathcal{A}^{p}_{\text{Dir,Bal}}$	$\mathcal{A}^{p}_{\text{Iter,Bal}}$	$\mathcal{A}^{p}_{\text{Dir,Lam}}$	$\mathcal{A}^{p}_{\text{Iter,Lam}}$	$\mathcal{A}^{p}_{\text{Dir,Con}}$	$\mathcal{A}^{p}_{\text{Iter,Con}}$
Quality Input Curves	$\mathbf{q}_1$	0.014	0.017	0.024	0.021	0.018	0.032	0.030
	$\mathbf{q}_2$	0.008	0.015	0.027	0.012	0.015	0.024	0.027
	$\mathbf{q}_3$	0.015	0.025	0.021	0.020	0.020	0.020	0.027
	$\mathbf{q}_4$	0.007	0.014	0.014	0.021	0.028	0.026	0.021
	$\mathbf{q}_5$	0.011	0.018	0.016	0.018	0.018	0.021	0.029
	$\mathbf{q}_6$	0.013	0.018	0.014	0.035	0.020	0.024	0.019
	$\mathbf{q}_7$	0.014	0.014	0.015	0.014	0.017	0.031	0.010
	$\mathbf{q}_8$	0.017	0.018	0.012	0.028	0.019	0.025	0.024
	$\mathbf{q}_9$	0.013	0.018	0.028	0.025	0.018	0.029	0.021
	$\mathbf{q}_{10}$	0.012	0.021	0.016	0.014	0.015	0.022	0.018
	$\mathbf{q}_{11}$	0.010	0.022	0.019	0.024	0.021	0.029	0.028

Table 5.5: Standard deviations of hypervolumes of solutions achieved for different input quality curves with different algorithms

		Algorithms				
		$\mathcal{A}^v_{\text{Mirr,Bal}}$	$\mathcal{A}^p_{\text{Dir,Bal}}$	$\mathcal{A}^p_{\text{Iter,Bal}}$	$\mathcal{A}^p_{\text{Dir,Lam}}$	$\mathcal{A}^p_{\text{Iter,Lam}}$
Population Sizes	**10**	0.208	0.170	0.172	0.137	0.168
	26	0.182	0.110	0.125	0.150	0.104
	60	0.167	0.098	0.098	0.105	0.118
	100	0.127	0.082	0.086	0.093	0.109
	150	0.133	0.081	0.112	0.069	0.093
	300	0.156	0.089	0.108	0.099	0.125
	500	0.194	0.117	0.115	0.150	0.150

Table 5.6: Set distances of median solutions testing the effect of different population sizes on the algorithms

		Algorithms				
		$\mathcal{A}^v_{\text{Mirr,Bal}}$	$\mathcal{A}^p_{\text{Dir,Bal}}$	$\mathcal{A}^p_{\text{Iter,Bal}}$	$\mathcal{A}^p_{\text{Dir,Lam}}$	$\mathcal{A}^p_{\text{Iter,Lam}}$
Population Sizes	**10**	0.048	0.048	0.051	0.061	0.040
	26	0.026	0.037	0.035	0.071	0.024
	60	0.032	0.035	0.053	0.042	0.049
	100	0.032	0.029	0.036	0.026	0.030
	150	0.034	0.036	0.028	0.023	0.041
	300	0.022	0.018	0.040	0.043	0.044
	500	0.019	0.041	0.035	0.059	0.045

Table 5.7: Standard deviations of set distances testing the effect of different population sized on the algorithms

its parents (crossover) with a certain probability and mutated. The population size is an indication for the stability of the development. The more individuals in a population the higher the stability of the development of the population is as the drop of an old individual has relatively more impact on a small population as on a huge population. Tables 5.6 and 5.7 show the median set distances and the according standard deviations for different population sizes and the described algorithms. As it can be seen a population size of 100 or 150 seems to give the best results in average for all algorithms. The set distance is the lowest and the standard deviation is acceptable whereas the highest population size of 500 and the lowest population size of 10 both give unstable and bad results compared to the results achieved with population sizes of 100 and 150.

5.8.2 Mutation and Crossover Probability

The mutation and crossover probabilities describe the likelihood of each operator to be applied to a child generated from the population. Table 5.8 and 5.9 show the effect of these probabilities on the results generated by the different algorithms in hypervolume measure and in set distance measure respectively.

The effect which can be observed in both tables (set distance is included for completeness, hypervolume for better pronunciation of the effect) is the high importance of the crossover algorithm using any of the algorithms. The best median results can be achieved with a crossover probability of 0.8 or 1.0 The lower the crossover probability is set, the higher the results rely on the value of the mutation probability. In average a mutation probability of 0.09 can be found as optimal for the selected problem. In contrast the best results are not achieved with the maximal mutation probability.

The tables also show the importance of the mutation probability. If set to zero the results are in general the worst achieved in these tests. This is easily explicable as new individuals are generated from the existing population and the crossover operator only recombines existing strategies. This is already one of the necessary steps for generation of variation but without the input of new types of parts for individuals through the mutation operator the population evolves much slower and is restricted to the strategic pieces generated at the beginning of the optimization.

5.8.3 Maximum Speed and Variable Count

This subsection investigates the influence of the maximum speed and the amount of variables on the optimization. At least the maximum speed analysis is more of theoretical importance than of practical because the maximum speed in a real world system is given by the system used. The practical significance might be found in analyzing which effect the reduction or increase of the maximum speed would have on the system before deciding to invest in new machines.

Tables 5.10 and 5.11 show again the median hypervolume and the standard deviation of the hypervolume for the selected algorithms and different maximum speeds. The step-like hypervolume increase is obvious as the maximum speed increases from 10 over 20 to 30 after which the values are nearly constant. It has to be kept in mind that the amount of variables for this analysis was set to 30 and the speed is given in amount of layers placeable with static stacking with maximum speed.

To investigate on the correlation a second test was carried out where the maximum speed was fixed at 20 again but the amount of variables was varied. The results of this test are shown in tables 5.12 and 5.13. Here again the step-like behavior of the hypervolume values around the variables count of 15 and 20 is observable.

The idea at this point is that the amount of variables should match the maximum speed. Following this idea another test was conducted which fixed varaibles and maximum speed to be equal and the results are shown in table 5.14. The table shows that the value increases in the beginning but seems to reach a saturation when the maximum speed and the amount of variables cross the values of 50.

These tests show an increase in maximum speed from 20 to higher values is beneficial for the optimization result (in the local setup) until it reaches the saturation at a speed of about 50. Furthermore the amount of variables should be at least as high as the maximum speed. An increase of the amount of variables to twice of the amount of layers gives even better results.

5.9 Mixed Repairing

Following the basic idea of Houck et al. [HJKW97] a mixed version of the Baldwinian and Lamarckian repairing was implemented where after the repairing and

		Mutation												
		0.000	**0.005**	**0.012**	**0.020**	**0.030**	**0.040**	**0.055**	**0.070**	**0.090**	**0.110**	**0.150**	**0.250**	**0.400**
$\mathcal{A}^{v}_{\text{Mirr,Bal}}$ Crossover	**0.0**	-	0.248	0.222	0.175	0.169	0.191	0.186	0.181	0.160	0.157	0.150	0.153	0.179
	0.2	0.257	0.206	0.183	0.186	0.199	0.162	0.143	0.146	0.132	0.155	0.139	0.156	0.147
	0.4	0.222	0.182	0.186	0.171	0.181	0.169	0.182	0.153	0.126	0.147	0.133	0.150	0.162
	0.6	0.206	0.169	0.175	0.176	0.158	0.160	0.139	0.140	0.153	0.147	0.139	0.153	0.174
	0.8	0.165	0.173	0.164	0.165	0.138	0.147	0.148	0.130	0.146	0.148	0.148	0.146	0.174
	1.0	0.153	0.152	0.155	0.135	0.147	0.127	0.135	0.139	0.118	0.144	0.153	0.159	0.161
$\mathcal{A}^{p}_{\text{Dir,Bal}}$ Crossover	**0.0**	-	0.257	0.232	0.149	0.144	0.133	0.103	0.108	0.107	0.106	0.091	0.077	0.112
	0.2	0.338	0.141	0.130	0.108	0.079	0.078	0.078	0.080	0.059	0.090	0.072	0.083	0.107
	0.4	0.236	0.119	0.078	0.078	0.104	0.083	0.083	0.077	0.094	0.081	0.078	0.070	0.115
	0.6	0.190	0.097	0.062	0.085	0.099	0.091	0.092	0.075	0.098	0.071	0.084	0.078	0.108
	0.8	0.189	0.104	0.098	0.083	0.074	0.088	0.068	0.069	0.068	0.092	0.066	0.084	0.108
	1.0	0.141	0.102	0.081	0.069	0.061	0.084	0.079	0.072	0.080	0.076	0.063	0.082	0.104
$\mathcal{A}^{p}_{\text{Iter,Bal}}$ Crossover	**0.0**	-	0.217	0.172	0.155	0.147	0.134	0.103	0.124	0.093	0.098	0.081	0.099	0.099
	0.2	0.327	0.164	0.151	0.103	0.123	0.126	0.093	0.088	0.070	0.084	0.083	0.086	0.111
	0.4	0.319	0.131	0.132	0.123	0.105	0.069	0.079	0.065	0.092	0.078	0.114	0.084	0.106
	0.6	0.225	0.122	0.094	0.101	0.094	0.100	0.091	0.087	0.102	0.084	0.118	0.098	0.109
	0.8	0.192	0.139	0.088	0.087	0.105	0.055	0.098	0.079	0.097	0.074	0.088	0.090	0.114
	1.0	0.170	0.144	0.120	0.094	0.107	0.091	0.070	0.083	0.085	0.081	0.083	0.078	0.099
$\mathcal{A}^{p}_{\text{Dir,Lam}}$ Crossover	**0.0**	-	0.226	0.187	0.163	0.149	0.152	0.127	0.119	0.085	0.079	0.089	0.101	0.099
	0.2	0.336	0.136	0.116	0.147	0.114	0.098	0.106	0.123	0.099	0.079	0.091	0.105	0.112
	0.4	0.252	0.121	0.162	0.097	0.107	0.086	0.082	0.095	0.080	0.073	0.087	0.089	0.109
	0.6	0.223	0.103	0.121	0.101	0.095	0.106	0.083	0.077	0.084	0.085	0.069	0.079	0.109
	0.8	0.184	0.124	0.114	0.100	0.113	0.079	0.079	0.085	0.077	0.081	0.090	0.109	0.115
	1.0	0.147	0.115	0.091	0.101	0.086	0.067	0.063	0.070	0.085	0.074	0.072	0.106	0.108
$\mathcal{A}^{p}_{\text{Iter,Lam}}$ Crossover	**0.0**	-	0.300	0.209	0.152	0.160	0.165	0.124	0.119	0.119	0.115	0.103	0.097	0.117
	0.2	0.314	0.148	0.134	0.129	0.118	0.117	0.094	0.098	0.097	0.103	0.110	0.112	0.139
	0.4	0.364	0.153	0.133	0.125	0.131	0.118	0.092	0.072	0.083	0.079	0.098	0.096	0.108
	0.6	0.290	0.127	0.120	0.108	0.127	0.088	0.093	0.094	0.089	0.075	0.084	0.102	0.107
	0.8	0.231	0.138	0.095	0.105	0.101	0.118	0.089	0.094	0.110	0.091	0.078	0.103	0.126
	1.0	0.206	0.152	0.123	0.098	0.095	0.096	0.094	0.091	0.096	0.085	0.086	0.106	0.104

Table 5.8: Set distances of median solutions showing the mutation and crossover parameter varied for the different algorithms

			Mutation												
			0.000	0.005	0.012	0.020	0.030	0.040	0.055	0.070	0.090	0.110	0.150	0.250	0.400
HSP1	Crossover	0.0	0.000	0.013	0.022	0.034	0.041	0.028	0.033	0.032	0.054	0.050	0.051	0.053	0.037
		0.2	0.007	0.034	0.038	0.034	0.019	0.050	0.051	0.050	0.057	0.056	0.056	0.051	0.046
		0.4	0.024	0.041	0.041	0.045	0.049	0.051	0.045	0.048	0.058	0.053	0.063	0.048	0.045
		0.6	0.033	0.046	0.036	0.042	0.057	0.060	0.052	0.057	0.052	0.060	0.054	0.052	0.044
		0.8	0.045	0.049	0.047	0.046	0.056	0.056	0.058	0.056	0.054	0.051	0.055	0.053	0.041
		1.0	0.053	0.061	0.052	0.060	0.056	0.066	0.068	0.060	0.066	0.061	0.051	0.047	0.049
HSP2	Crossover	0.0	0.000	0.000	0.019	0.033	0.050	0.046	0.077	0.060	0.068	0.072	0.080	0.084	0.057
		0.2	0.000	0.044	0.060	0.077	0.080	0.088	0.082	0.080	0.096	0.079	0.091	0.077	0.071
		0.4	0.002	0.057	0.081	0.088	0.071	0.084	0.095	0.090	0.081	0.083	0.083	0.092	0.060
		0.6	0.032	0.075	0.089	0.081	0.071	0.085	0.073	0.091	0.069	0.093	0.084	0.084	0.066
		0.8	0.019	0.064	0.073	0.073	0.083	0.080	0.091	0.088	0.087	0.077	0.088	0.082	0.061
		1.0	0.050	0.090	0.090	0.095	0.103	0.087	0.090	0.088	0.096	0.085	0.095	0.089	0.066
HSP3	Crossover	0.0	0.000	0.000	0.029	0.034	0.038	0.050	0.062	0.061	0.069	0.077	0.090	0.065	0.064
		0.2	0.000	0.027	0.039	0.075	0.057	0.056	0.070	0.085	0.086	0.074	0.080	0.086	0.059
		0.4	0.000	0.041	0.063	0.061	0.072	0.089	0.092	0.095	0.080	0.087	0.073	0.077	0.067
		0.6	0.010	0.062	0.076	0.062	0.069	0.075	0.073	0.083	0.069	0.086	0.053	0.066	0.069
		0.8	0.019	0.046	0.072	0.082	0.070	0.101	0.075	0.079	0.076	0.081	0.078	0.079	0.059
		1.0	0.033	0.078	0.066	0.069	0.082	0.079	0.096	0.093	0.087	0.097	0.084	0.083	0.073
HSP4	Crossover	0.0	0.000	0.003	0.025	0.031	0.055	0.040	0.056	0.076	0.085	0.084	0.079	0.064	0.068
		0.2	0.000	0.038	0.057	0.048	0.068	0.073	0.066	0.057	0.074	0.089	0.079	0.072	0.064
		0.4	0.002	0.055	0.031	0.066	0.055	0.082	0.088	0.074	0.082	0.087	0.078	0.073	0.070
		0.6	0.006	0.064	0.060	0.077	0.076	0.076	0.080	0.088	0.086	0.074	0.089	0.081	0.071
		0.8	0.018	0.071	0.069	0.076	0.054	0.083	0.085	0.080	0.087	0.094	0.093	0.066	0.069
		1.0	0.050	0.064	0.080	0.078	0.088	0.094	0.101	0.095	0.091	0.088	0.104	0.059	0.056
HSP5	Crossover	0.0	0.000	0.000	0.019	0.039	0.033	0.029	0.060	0.067	0.067	0.071	0.074	0.071	0.050
		0.2	0.000	0.037	0.043	0.051	0.062	0.067	0.082	0.062	0.077	0.074	0.057	0.058	0.043
		0.4	0.000	0.037	0.054	0.038	0.066	0.064	0.070	0.091	0.079	0.075	0.067	0.078	0.059
		0.6	0.000	0.046	0.064	0.065	0.055	0.079	0.080	0.074	0.077	0.084	0.089	0.067	0.058
		0.8	0.006	0.050	0.070	0.065	0.068	0.059	0.084	0.085	0.062	0.072	0.085	0.076	0.042
		1.0	0.015	0.050	0.047	0.072	0.083	0.073	0.075	0.081	0.073	0.086	0.069	0.076	0.072

Table 5.9: Hypervolumes of median solution showing the mutation and crossover parameter varied for the different algorithms

		Algorithms		
		$\mathcal{A}^v_{\text{Mirr,Bal}}$	$\mathcal{A}^p_{\text{Dir,Bal}}$	$\mathcal{A}^p_{\text{Iter,Bal}}$
Maximum Speeds	**10**	0.009	0.060	0.032
	20	0.062	0.083	0.090
	30	0.073	0.100	0.099
	40	0.073	0.098	0.113
	50	0.077	0.102	0.115
	60	0.078	0.099	0.105
	70	0.073	0.105	0.107
	80	0.071	0.105	0.103

Table 5.10: Median hypervolumes of solutions sets calculated for varying maximum speeds with different algorithms

		Algorithms		
		$\mathcal{A}^v_{\text{Mirr,Bal}}$	$\mathcal{A}^p_{\text{Dir,Bal}}$	$\mathcal{A}^p_{\text{Iter,Bal}}$
Maximum Speeds	**10**	0.013	0.018	0.022
	20	0.014	0.017	0.024
	30	0.005	0.016	0.019
	40	0.008	0.014	0.018
	50	0.008	0.019	0.015
	60	0.009	0.022	0.017
	70	0.009	0.009	0.015
	80	0.003	0.010	0.026

Table 5.11: Standard Deviation of hypervolumes of solutions sets calculated for varying maximum speeds with different algorithms

		Algorithms		
		$\mathcal{A}^v_{\text{Mirr,Bal}}$	$\mathcal{A}^p_{\text{Dir,Bal}}$	$\mathcal{A}^p_{\text{Iter,Bal}}$
Variables	**5**	0.061	0.000	0.000
	10	0.060	0.002	0.000
	15	0.059	0.016	0.030
	20	0.064	0.084	0.083
	25	0.065	0.079	0.069
	30	0.062	0.083	0.090
	40	0.053	0.100	0.095
	50	0.056	0.080	0.097
	60	0.049	0.101	0.088
	80	0.026	0.099	0.084
	100	0.015	0.069	0.068

Table 5.12: Median hypervolumes of solutions sets calculated for varying the amount of variables with different algorithms

		Algorithms		
		$\mathcal{A}^v_{\text{Mirr,Bal}}$	$\mathcal{A}^p_{\text{Dir,Bal}}$	$\mathcal{A}^p_{\text{Iter,Bal}}$
Variables	**5**	0.003	0.000	0.000
	10	0.005	0.003	0.001
	15	0.008	0.012	0.020
	20	0.004	0.023	0.017
	25	0.010	0.013	0.020
	30	0.014	0.017	0.024
	40	0.012	0.013	0.023
	50	0.013	0.015	0.024
	60	0.027	0.014	0.020
	80	0.018	0.011	0.032
	100	0.014	0.017	0.068

Table 5.13: Standard deviations of hypervolumes of solutions sets calculated for varying the amount of variables with different algorithms

		Algorithm $\mathcal{A}^{p}_{\text{Dir,Bal}}$
Variables / Max. Speed	**10 / 10**	0.000
	20 / 20	0.084
	30 / 30	0.100
	40 / 40	0.105
	50 / 50	0.117
	60 / 60	0.110
	70 / 70	0.120
	80 / 80	0.120
	100 / 100	0.118
	120 / 120	0.121
	140 / 140	0.115

Table 5.14: Median hypervolumes of solution sets calculated while scaling variables and maximum speed simultaneously

the evaluation a repaired individual would be written back to the population with a specific probability. The good effect of joining the benefits of both algorithmic decisions could not be confirmed for this problem, as table 5.15 confirms. The table mainly shows the nearly steady trend of the results from Baldwinian to Lamarckian repairing application as the results are the best repairing the Baldwinian way and the worst results are achieved the more Lamarckian repairing is applied.

5.10 Feasibility Development

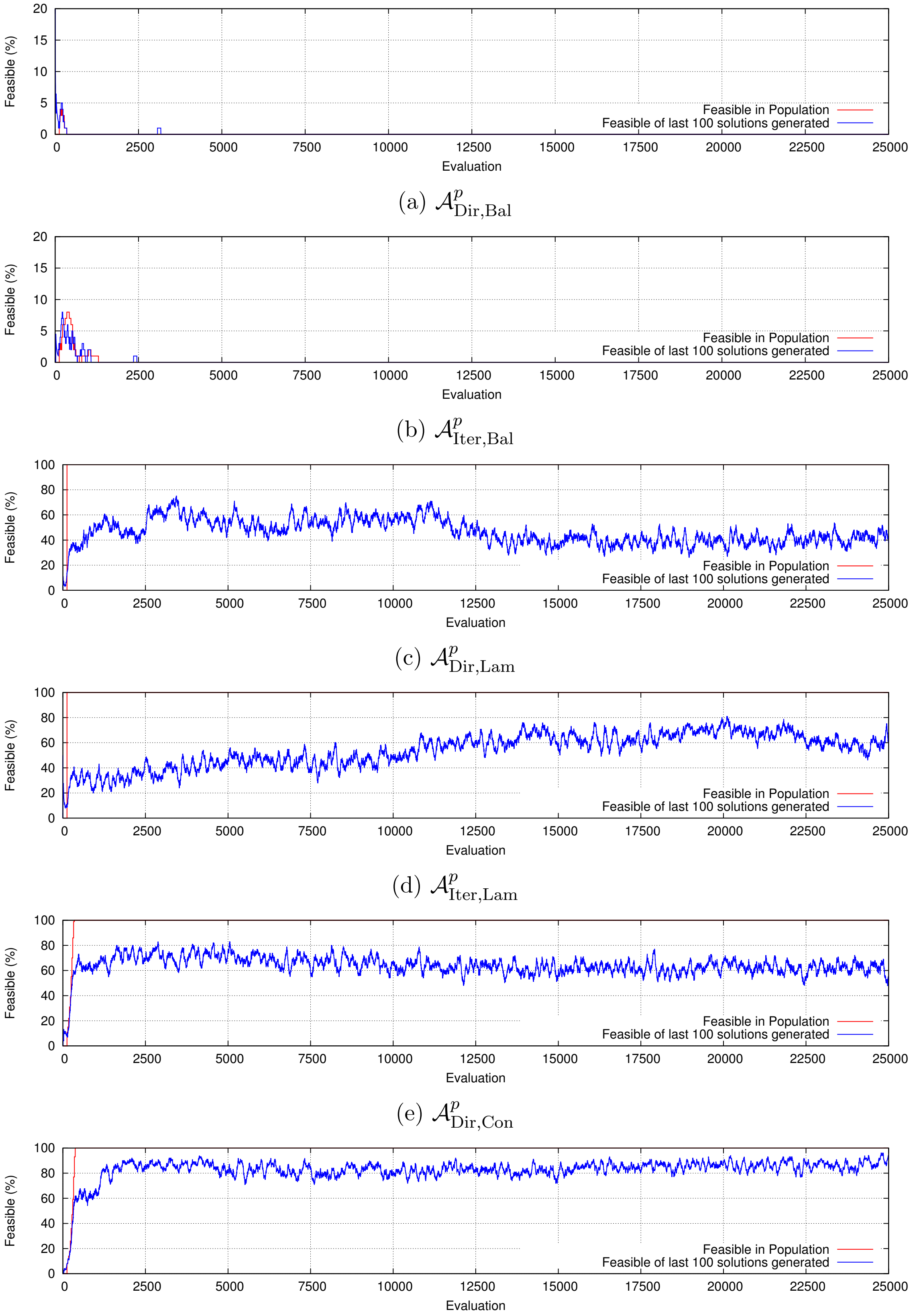
This section analyzes the effect of the optimization algorithms on the feasibility development. Figures 5.18 show the feasibility development the populations during the optimization process together with the generation of feasible child solutions.

The optimization using the Baldwinian repairing optimization starts with some feasible results (generated randomly) but these results are quickly dropped and no feasible results appear during the later optimization process. Other runs produces similar results as this can be taken as a proof that the objective optimization itself does not drive the solutions to feasibility.

The runs with the Lamarckian repairing show an immediate jump to a fully feasible population as all individuals are repaired and written back to the population. Newly generated children benefit from this repairing feasibility as the chance for them to be feasible is around 50% for an average run.

Using the constrain measure interestingly also drives the population to 100% feasible within only several hundred evaluations and shows in average a higher stability for the feasibility of newly generated child solutions of 70% to 80%.

In the curves shown in figure 5.19 the feasibility development using the mixed repairing with different probabilities for Baldwinian and Lamarckian repairing is depicted. Interestingly the amount of newly generated feasible solutions increases quickly if



(a) $\mathcal{A}^{p}_{\text{Dir,Bal}}$

(b) $\mathcal{A}^{p}_{\text{Iter,Bal}}$

(c) $\mathcal{A}^{p}_{\text{Dir,Lam}}$

(d) $\mathcal{A}^{p}_{\text{Iter,Lam}}$

(e) $\mathcal{A}^{p}_{\text{Dir,Con}}$

(f) $\mathcal{A}^{p}_{\text{Iter,Con}}$

Figure 5.18: Analysis of generation of feasible solutions using different algorithms

		Hypervolume		Set Distance	
		Value	Std. Dev.	Value	Std. Dev.
Lamarckian / Baldwinian	0%/100%	0.085	0.020	0.076	0.029
	5%/95%	0.077	0.021	0.094	0.031
	20%/80%	0.071	0.022	0.092	0.033
	35%/65%	0.074	0.020	0.093	0.030
	50%/50%	0.068	0.024	0.102	0.041
	65%/35%	0.062	0.022	0.110	0.030
	80%/20%	0.062	0.022	0.102	0.038
	95%/5%	0.070	0.023	0.101	0.037
	100%/0%	0.067	0.026	0.102	0.043

Table 5.15: Mixed Lamarckian vs. Baldwinian Repairing with $\mathcal{A}^{p}_{\text{Iter,Mix}}$

only a small part of 5% part of the repaired solutions is being written back to the population. But if the amount is increased further the results converge to the ones described for the Lamarckian repairing application.

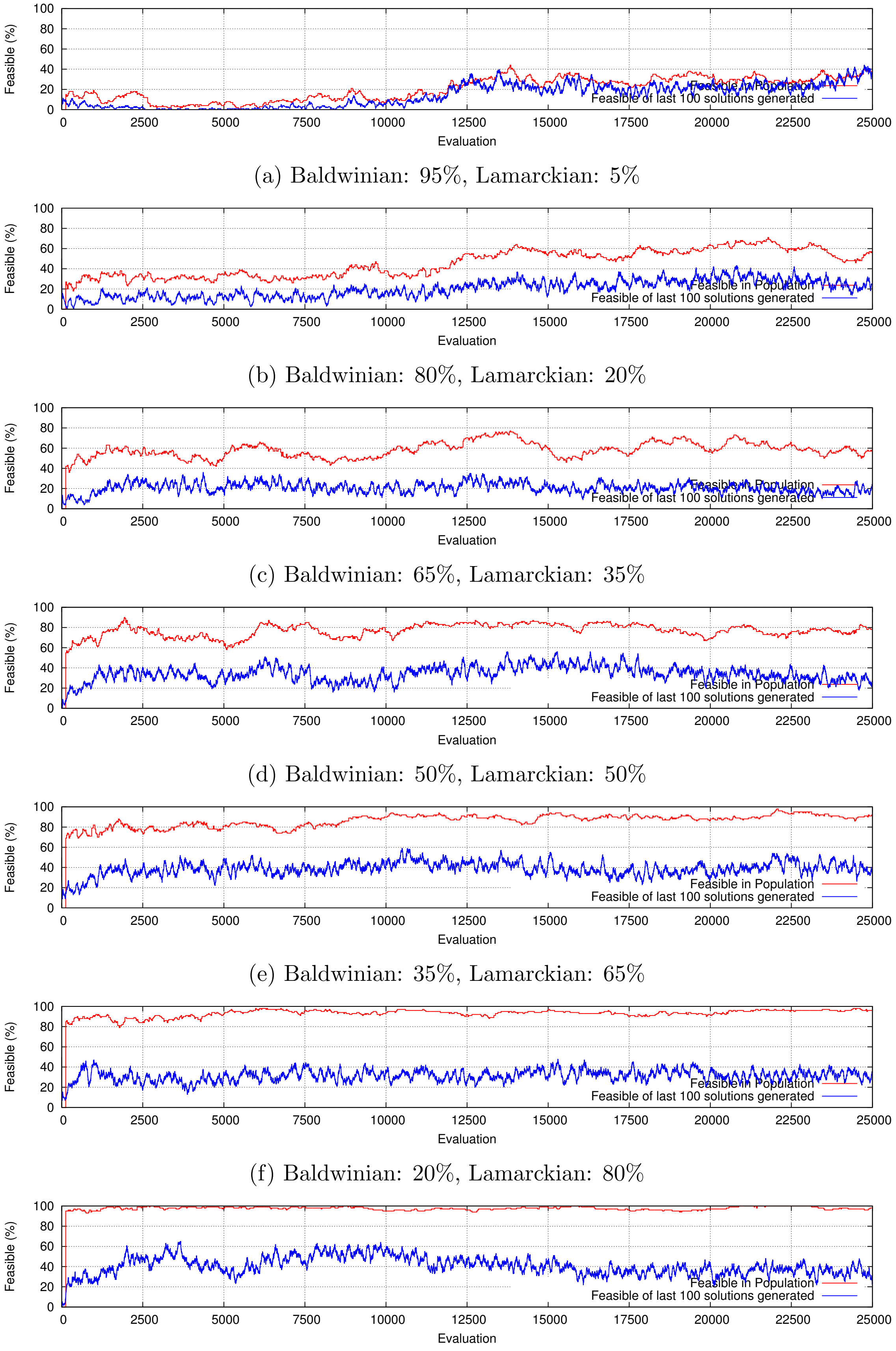
5.11 Summary

This chapter presented the main results achieved with the optimization system. During the explanation of the intermediate results the intuitively understandable results for the stacker traverse path solutions were shown. Also the continuously improving development of the solutions during an optimization run was illustrated. The solutions found by the algorithms always outperform the results achieved with the static stacking, whose further disadvantages were also depicted.

The solutions found in the non-dominated set used for distance measure gave highly interesting insights about the algorithms used, as they show which algorithms are the most useful for calculating specific parts of the non-dominated set. The algorithms $\mathcal{A}^{p}_{\text{Dir,Bal}}$and $\mathcal{A}^{p}_{\text{Dir,Lam}}$could be found as the most valuable for optimization. Also a strongly pronounced knee region could be identified in the solution sets found during the calculations.

The parameter variation gave insights about the behavior of the algorithms with different parameter sets. The crossover probability could be identified as most important for a good population development with at least a minimal mutation probability. A population size of 100 to 150 individual solutions showed the best optimization results. A strong correlation between maximum speed and amount of variables could be presented and a saturation effect for the combined scaling was shown.

The results calculated with the mixed Lamarckian and Baldwinian repairing algorithm $\mathcal{A}^{p}_{\text{Iter,Mix}}$do not benefit from the partial Lamarckianism but it was observed that a little repairing with writing back to the population already has a huge effect on the generation of feasible solutions.



(a) Baldwinian: 95%, Lamarckian: 5%

(b) Baldwinian: 80%, Lamarckian: 20%

(c) Baldwinian: 65%, Lamarckian: 35%

(d) Baldwinian: 50%, Lamarckian: 50%

(e) Baldwinian: 35%, Lamarckian: 65%

(f) Baldwinian: 20%, Lamarckian: 80%

(g) Baldwinian: 5%, Lamarckian: 95%

Figure 5.19: Analysis of generation of feasible solutions using different $\mathcal{A}^{p}_{\mathrm{Dir,Mix}}$

It can be concluded that the presented model and the algorithms are suitable for solving the given problem of optimizing the blending performance for a given quality input curve. Furthermore the results allow the derivation of knowledge about the algorithms used for optimization and about the strategies for optimizing the described objectives.

6. Summary and Future Work

The problem of optimizing the blending performance of a longitudinal blending system in bulk material processing was described in detail. The main variable for optimization could be found in the stacking machine traverse path. The main objectives were described as the low quality variance in the material output stream and the near-constant height of the stockpile.

The problem was modeled using a particle based simulation of the stockpile where each particle has been assigned a specific quality from the current input quality curve. The simulation allows the virtual ablation of the stockpile and thus the generation of a quality output curve. This is used for the evaluation of a possible solution for a specific problem instance.

The objectives together with the additional constraints like maximum speed and stockpile limits were considered when defining the algorithm operators and mechanisms for evolution and repairing. Several algorithmic combinations were defined for in-depth analysis of their effects on the optimization problem.

The algorithms used were found as highly suitable for the given multi-objective optimization problem. The results prove a reliable and good optimization of the Pareto front which describes the specific best solution set of each optimization run. The comparison of generated solution sets using the hypervolume measure and set distance to the non-dominated set allows the comparison of the algorithms.

Several parameter combinations could be found as the most promising for an optimization run but furthermore the results generated during the calculations allow insights into the algorithms and into the optimization problem. On one hand the solutions describe intuitively understandable traverse paths and on the other hand one gets findings about the algorithmic behavior for specific parts of the optimization problem. Thus the direct repairing in the Baldwinian way could be found as the most stable for the optimization. But also other algorithms are of high value for generation of results beyond the knee region and at extreme points.

The presented work is an ongoing research with high interest in the industry as it offers the optimization of existing blending beds without the need expensive hardware changes. The quality measurement combined with an adequate software implementation of an optimization algorithm offers the possibility of optimization with low financial and hardware effort.

The next step of this development is surely the further improvement of the optimization system to be even more reliable and to provide a high guarantee for good solution set generation within shorter calculation time. This can be done, for instance, by restriction on predefined positions, experimenting with dynamic time slots, use of a specific knee finding algorithm or starting with a predefined first population generation.

Other interesting results could be found using the detailed simulation for the optimization. This will be possible in near future as the calculation power of personal computers is increasing strongly with regard to parallel computation on graphics cards. This new computation potential allows a much faster computation of the real world physics simulation and thus a more realistic simulation.

Offering a real-time optimization system for application in a bulk material processing plant using the presented system is a goal set for the next years and will be researched with high effort as the results presented are very encouraging.

Bibliography

[Ald12] J.K. Alderman. *Coal Blending and Optimization*, pages 52–62. Coal Age, 2012.

[BCW00] J.E. Bond, R. Coursaux, and R.L. Worthington. Blending systems and control technologies for cement raw materials. *IEEE Industry Applications Magazine*, 6:49–59, 2000.

[Beu12] N. Beume. *Hypervolume based metaheuristics for multiobjective optimization.* PhD thesis, Eldorado, TU Dortmund, January 2012.

[bul13] Bullet physics library. http://bulletphysics.org, April 2013.

[Cou13] E. Coumans. Gpu rigid body simulation. Technical report, Game Developers Conference, 2013.

[CSB$^+$12] M.P. Cipold, P.K. Shukla, C.C. Bachmann, K. Bao, and H. Schmeck. An evolutionary optimization approach for bulk material blending systems. In C.A. Coello Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, editors, *PPSN (1)*, volume 7491 of *Lecture Notes in Computer Science*, pages 478–488. Springer, 2012.

[Deb01] K. Deb. *Multi-objective optimization using evolutionary algorithms.* Wiley, 2001.

[DG11] K. Deb and S. Gupta. Understanding knee points in bicriteria problems and their implications as preferred solution principles. *Engineering Optimization*, 43(11):1175–1204, 2011.

[DNLA09] J.J. Durillo, A.J. Nebro, F. Luna, and E. Alba. On the effect of the steady-state selection scheme in multi-objective genetic algorithms. In M. Ehrgott, C.M. Fonseca, X. Gandibleux, J. Hao, and M. Sevaux, editors, *Evolutionary Multi-Criterion Optimization*, volume 5467 of *Lecture Notes in Computer Science*, pages 183–197. Springer Berlin Heidelberg, 2009.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[HJKW97] C.R. Houck, J.A. Joines, M.G. Kay, and J.R. Wilson. Empirical investigation of the benefits of partial lamarckianism. *Evolutionary Computation*, 5(1):31–60, 1997.

[jme13] jmetal - a framework for multi-objective optimization. http://jmetal.sourceforge.net, April 2013.

[Kum03] M. Kumral. Bed blending design incorporating multiple regression modelling and genetic algorithms. *International Journal of Surface Mining, Reclamation and Environment*, 17:98–112, 2003.

[LB04] M.J. Laurila and C.C. Bachmann. X-ray fluorescence measuring system and methods for trace elements. *US Patent US 2004/0240606*, 2004.

[PA06] F.F. Pavloudakis and Z. Agioutantis. Simulation of bulk solids blending in longitudinal stockpiles. *Journal of the South African Institute of Mining and Metallurgy*, 106:229–237, 2006.

[Pet04] I.F. Petersen. Blending in circular and longitudinal mixing piles. *Chemometrics and Intelligent Laboratory Systems*, (74):135–141, 2004.

[rec13] Image of a stacking machine. https://en.wikipedia.org/wiki/File:Krupp_stacker_rtca_kestrel_mine.jpg, April 2013.

[SBS13] P.K. Shukla, M.A. Braun, and H. Schmeck. Theory and algorithms for finding knees. In R.C. Purshouse, P.J. Fleming, C.M. Fonseca, S. Greco, and J. Shaw, editors, *EMO*, volume 7811 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2013.

[SPR11] A.K. Swain, H. Patra, and G.K. Roy. *Mechanical Operations, 1E*. McGraw-Hill Education (India) Pvt Limited, 2011.

[sta13] Image of a reclaiming machine. https://commons.wikimedia.org/wiki/File:Krupp_bridge_reclaimer_rtca_kestrel_mine.jpg, April 2013.