An Evolutionary Optimization Approach for Bulk Material Blending Systems

Michael P. Cipold[1,2], Pradyumn Kumar Shukla[1], Claus C. Bachmann[2], Kaibin Bao[1], and Hartmut Schmeck[1]

[1] Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, D-76128, Germany
[2] J&C Bachmann GmbH, Bad Wildbad, D-75323, Germany

Abstract. Bulk material blending systems still mostly implement static and non-reactive material blending methods like the well-known Chevron stacking. The optimization potential in the existing systems which can be made available using quality analyzing methods as online X-ray fluorescence measurement is inspected in detail in this paper using a multi-objective optimization approach based on steady state evolutionary algorithms. We propose various Baldwinian and Lamarckian repair algorithms, test them on real world problem data and deliver optimized solutions which outperform the standard techniques.

Keywords: Bulk Material Blending, Multi-objective Evolutionary Algorithms, Chevron Stacking.

1 Introduction

Naturally occurring materials like minerals, coals, and ores are inherently inhomogeneous products. However, efficient processing of those products requires that the quality of the product does not vary beyond a limited range. While a constant quality cannot be assured during mining or refining the uniform quality can be achieved by measuring the quality and blending the material.

A whole range of bulk material blending systems are developed to buffer bulk material from a source, blend it and finally deliver a product with homogenized quality parameters. As quantity of bulk material is usually measured in ten thousands of tons, the implemented systems are using huge machines to process the material in so-called blending beds. Blending beds are areas of typical length up to 1000m and width up to 50m which can be used by the stacking and reclaiming machines to build and ablate stockpiles and can be organized in multiple ways [1]. In this paper, the focus lies on longitudinal blending beds that use a railed stacker (shown in Figure 1) and a railed bridge reclaimer. Figure 2 illustrates a blending bed with all elements as it is used for the model

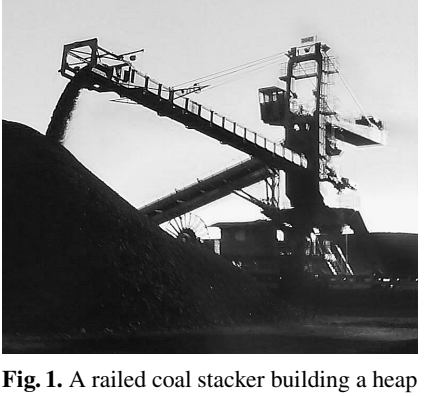

Fig. 1. A railed coal stacker building a heap

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 478–488, 2012.

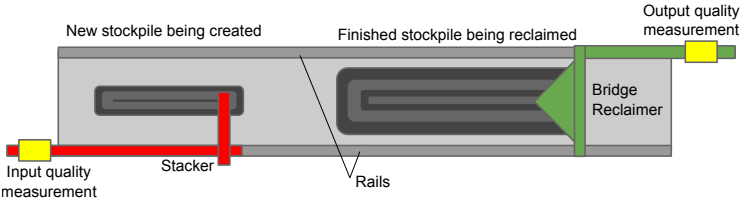

Fig. 2. Top view sketch of a blending bed with one heap being stacked and another heap being reclaimed

in this paper using simplified parameter and environment descriptions to concentrate on the main goal, i.e., to deliver a homogeneous product.

The homogenization in blending beds is usually done by stacking layers of material onto each other and reclaiming the material vertically or diagonal to the layers to get mixed material from the cross sections through the layers. As the amount of layers is set to a fixed value at the beginning of building a stockpile, in the most cases the variation of the material parameters (for the remaining part of the paper referred with the abstract description *quality*) is not homogenized in an optimal way. Most established blending systems do not really use the optimization potential what in turn results in inefficient downstream processing. As these blending systems can not be modified in general without putting in much effort and money, this paper is aiming to analyze the optimization potential with minimal intrusion to a given system by using the possibilities which inhere in such a system: spreading the material in a dynamic way according to a quality analyzing online X-ray fluorescence measurement [2].

To analyze the maximum efficiency of a blending system we require to know the full quality of the material input (hereinafter referenced as *quality input curve*) from the beginning of the optimization. In this paper we present an optimization environment which is based on this information and calculates an optimized material spread by modifying the traverse path for the stacker. The approach is also portable to blending systems with other parameter and design choices.

Many authors have analyzed and improved the blending efficiency in the field of bulk material blending. Kumral [3] describes 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 use a more complex stockpile simulation approach consisting of multiple layers lying on top of each other [4]. They assume a constant material flow and calculate the expected quality at the material output. Bond et. al. [1] describe methods to improve the efficiency of blending beds by dynamically modifying the volume of a stockpile and modifying the stacking speed while measuring the input quality with an online analyzer. They 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. The optimization based approach has, to the best of our knowledge, never been applied in such a system and this is the topic dealt in this paper.

This paper is divided into five sections of which this is the first. The next section describes the problem and a simulation model that is developed. Section 3 presents

individual representations and repair methodologies that are used in a steady-state multi-objective evolutionary algorithm. The optimization results are discussed in Section 4 and conclusions are presented in Section 5.

2 Problem Modeling

The described real world system consists of multiple elements which need to be mapped into a system model first. The model consists of:

- *blending bed parameters* **b** (fixed parameters such as the stockpile dimensions, maximum stacker moving speed, reclaimer angle, etc.),
- *quality input* **q** (the quality curve for each run plotted against the total amount of material),
- *traverse path* **p** (the path along which the stacker is driving during the stockpile creation process), and
- *environment parameters* **e** (material and weather conditions).

Given that the stacking system is unchangeable and that the quality input is fixed for each run, the optimization focuses on modifying the stacker driving path along the blending bed. For simplicity, the environment parameters are not considered here. Hence, values of **b** and **q** define an instance of the optimization problem. In order to apply optimization with varying input data, the system was mapped into a detailed system model. The system uses a physics simulation to calculate the behavior of particles of various size being dropped onto a stockpile. Each of these particles is assigned a quality according to the current average quality being measured (or to an earlier recorded quality curve). The simulation system calculates the full stockpile as it is created in the real world including the effects of particles slipping from the steep sides of the stockpile. After building the stockpile the reclaiming is simulated by calculating the cross section quality average in the angle the reclaimer would ablate the material. One function evaluation (heap length 300m, heap width 40m) using the detailed simulator, with one million particles takes about 20 hours on an Intel Core 2 Duo, 2.50 GHz.

As the time for stacking one stockpile usually varies between 24 and 48 hours, this near real time calculation is not useful for optimization where the quality output has to be calculated for various input paths. For the fast evaluation of a calculated solution another simplified simulation system is developed which is able to do a rough output quality calculation within 20-50 milliseconds on the same machine. The physics simulation is now replaced by a three-dimensional grid where each particle is dropped in a specific place and then it falls down until it reaches a resting particle. From this place, it falls into the direction of biggest height difference. A particle can fall in one of the eight directions (top-left, top-center, top-right, left-middle, right-middle, bottom-left, bottom-center, bottom-right) or, it can stay in its place if there is no direction with lower height to fall to. The amount of particles here was reduced to 3 particles per cubic meter to minimize the necessary amount of calculations. The results of this (simplified) simulator show a satisfying correlation with the results from the detailed simulation (see Figures 3 and 4, the Pearson's correlation ranges from 0.85 for low variance solutions and up to 0.99 for high variance solutions). Hence, this simulator can be used for a rough calculation of the optimized traverse path quality output in a short time.

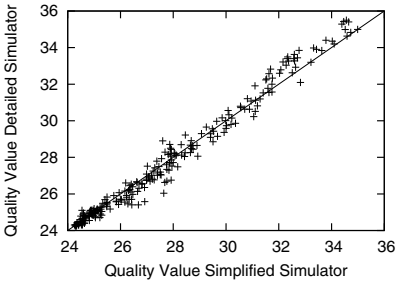

Fig. 3. Scatter plot showing high variance output quality correlation for detailed simulator against simplified simulator

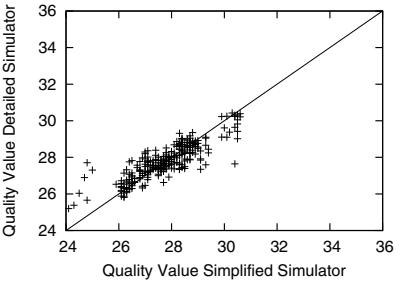

Fig. 4. Scatter plot showing low variance output quality correlation for detailed simulator against simplified simulator

Input Data. Input data was collected using an online X-ray fluorescence system delivering the material parameters of main interest in real time. This system was installed in a coal processing plant in South Africa. To gather more data for the simulation with other variation characteristics the recorded data was analyzed and replicated. The replicated data values are set in the same quality range but show a different variation timing to test the stability of the optimization system in situations with long constant quality or fast quality variation. For this, four different input quality curves with different characteristics will be used of which curve 1 is the original data from the coal processing plant. The problem corresponding to the first quality curve is termed as P1, and so on.

Objectives and Constraints. As described earlier, the traverse path $\mathbf{p}$ of the stacker has to be optimized for ideal material blending. The first objective here is the weighted variance of the quality at the reclaimer which is defined as

$$F_1(\mathbf{p}) := \frac{\tilde{n}(\mathbf{p}) \cdot \sum_{i=1}^{n} \left(w_i(\mathbf{p}) \cdot (q_i^{\text{out}}(\mathbf{p}) - \overline{q}^{\text{out}})^2 \right)}{\tilde{n}(\mathbf{p}) \cdot \sum_{i=1}^{n} w_i(\mathbf{p}) - \sum_{i=1}^{n} w_i(\mathbf{p})}, \tag{1}$$

where n is the number of cross sections, $w_i(\mathbf{p}), q_i^{\text{out}}(\mathbf{p})$ are the amount of material and average quality in cross section i, $\overline{q}^{\text{out}}$ is the average output quality (depending on the quality input $\mathbf{q}$), and $\tilde{n}(\mathbf{p})$ is the number of non-empty cross-sections. When the traverse path is changed it has a direct influence on the shape of the stockpile. This leads to the other primary objective to create a stockpile with a ridge of nearly constant height. The objective will be represented with the relative height difference defined as

$$F_2(\mathbf{p}) := \frac{h_{\max}(\mathbf{p}) - h_{\min}(\mathbf{p})}{\overline{h}(\mathbf{p})}, \tag{2}$$

where $h_{\max}(\mathbf{p}), h_{\min}(\mathbf{p})$ and $\overline{h}(\mathbf{p})$ denote the maximum, minimum, and the average stockpile heights respectively. Other objectives like having the least speed changes or driving only as fast as necessary to meet a defined threshold will not be regarded in

this paper but can be easily included in the method if required. The constraints in this optimization problem are:

- the stacker must traverse within the region $[0, p_{\max}]$ and
- the given speed range $[-v_{\max}, v_{\max}]$ may not be violated,

where $p_{\max}$ and $v_{\max}$ are the maximal length of the blending bed and the maximum speed of the stacker, respectively (the stacker moves along the rails in two directions).

3 Algorithms

The bi-objective problem described in the last section cannot be written in a closed mathematical form, prohibiting the use of exact algorithms (like [5]). Due to this, we used a steady-state version of NSGA-II (ssNSGAII) [6, 7]. For all the four problems, we use a population of size 100 and set the maximum number of function evaluations as 25,000 (250 generations). We use a standard real parameter SBX and polynomial mutation operator with $\eta_c = 15$ and $\eta_m = 20$, respectively [6]. The individual representation and the repairing methods are explained next.

3.1 Representation of Individuals

In order to map the complex traverse path of a stacker to an individual of the evolutionary algorithm (and vice versa) specific representations are defined. For this, the continuous traverse path is discretized and two representations are proposed.

Array of Speeds. This path representation uses an array of l floating point values v_i which describes the stacker driving speed in a specific time slot. To have a universal representation for the algorithm the value range is set to $[-1, 1]$. A value of 0 represents no movement and an absolute value of 1 represents movement with maximum speed. The direction of the movement is encoded in the signum of the value. Hence, an individual representation is $(v_1, \ldots, v_l)$ where $v_i \in [-1, 1]$ for all i. The length of a time slot t_{slot} is defined by the amount of time which is needed for all material to be stacked t_{total} divided by the length of the array l, i.e., $t_{\text{slot}} = \frac{t_{\text{total}}}{l}$.

Example 1. Individual $(0.5, -1.0, 0.5)$ represents a stacker movement to the right with half of the maximum speed, then full speed movement to the left and again half speed movement to the right. Each time slot is one third of the full stacking time. □

Array of Positions. This path representation also utilizes an array of l floating point values p_i corresponding to positions in the valid traverse path of the stacker. The value range $[0, 1]$ corresponds to the full traverse range. Hence, an individual representation is $(p_1, \ldots, p_l)$ where $p_i \in [0, 1]$ for all i. The time for moving between two positions t_{slot} is calculated similar to the time at the *array of speeds* but as we have a specific starting point for each individual the value of t_{slot} slightly increases. In this case, $t_{\text{slot}} = \frac{t_{\text{total}}}{l-1}$.

Example 2. Individual $(0.5, 1.0, 0.0)$ represents a stacker movement from the center of the valid traverse path to the right end and then to the left end with each time slot being half of the maximum time. □

3.2 Repair Mechanisms

The *array of speeds* representation was chosen in the first place to have a simple representation where every created individual can be mapped to a valid traverse path using a repair method. The *array of positions* representation was chosen with regard to be able to repair the individuals within the optimization and not only for representation. In the following paragraphs the used repair methods for the representations will be described in detail.

Array of Speeds. Calculating the traverse path for the *array of speeds* representation obviously results in most individuals being invalid because the calculated path exceeds the limits. Therefore, a simple repair mechanism was defined using a mirroring technique. As the integration of the speed over time results in the absolute position p_{abs} the position can be folded into the valid range to p_{mirrored} using equation (3):

$$p_{\text{mirrored}} = \begin{cases} p_{\text{abs}} - \lfloor p_{\text{abs}} \rfloor, & \text{if } \lfloor p_{\text{abs}} \rfloor \bmod 2 = 0, \\ 1.0 - (p_{\text{abs}} - \lfloor p_{\text{abs}} \rfloor), & \text{if } \lfloor p_{\text{abs}} \rfloor \bmod 2 = 1. \end{cases} \tag{3}$$

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 speed changes. Mapping back the mirroring would mean to change the amount of variables and the fixed time between two speed changes to be set.

Example 3. Let $v_{\max} = \frac{2 \cdot \textit{width of stockpile}}{t_{\text{total}}}$ and the individual be $(1.0, 0.5)$ which corresponds to the maximum speed being as much as necessary to traverse the full stockpile width twice. Furthermore we assume the start position as absolutely left. Going with the first speed 1.0 will result in the stacker being at the right end of the stockpile as $t_{\text{slot}} = 0.5 \cdot t_{\text{total}}$. In the second part the absolute position still increases with half of the maximum speed until the absolute position value of 1.5 is reached. With $\lfloor 1.5 \rfloor \bmod 2 = 1$ the mirrored position results in $p_{\text{mirrored}} = 1.0 - (1.5 - \lfloor 1.5 \rfloor) = 1.0 - (1.5 - 1.0) = 0.5$. □

Array of Positions. Due to the definition of this representation it is not possible to exceed the path limits (cf. for the *array of speeds* where it was not possible to exceed the speed limit) but it is possible that the speed $v_i = \frac{p_{i+1} - p_i}{t_{\text{slot}}}$ between two consecutive positions p_i and p_{i+1} exceeds the maximum allowable speed $v_{\max}$. For this case two repair mechanism will be used which utilize the maximum position difference $d_{\max} = t_{\text{slot}} \cdot v_{\max}$:

– *Direct Correction* iterates through the array once and limits each following position to the maximum difference to its predecessor:

$$p_{i+1} = \begin{cases} p_i - d_{\max}, & \text{if } p_i - p_{i+1} > d_{\max}, \\ p_i + d_{\max}, & \text{if } p_{i+1} - p_i > d_{\max}, \\ p_{i+1}, & \text{otherwise,} \end{cases} \qquad \text{with } i = 1, \ldots, n-1. \tag{4}$$

– *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_{corr} which has to be corrected:

$$(p_i, p_{i+1}) = \begin{cases} (p_i - d_h, p_{i+1} + d_h), & \text{if } p_i - p_{i+1} > d_{\max}, \\ (p_i + d_h, p_{i+1} - d_h), & \text{if } p_{i+1} - p_i > d_{\max}, \\ (p_i, p_{i+1}), & \text{otherwise,} \end{cases} \tag{5}$$

$$\text{with} \quad i = 1, \ldots, n-1 \quad d_{\text{corr}} = |p_i - p_{i+1}| - d_{\max} \quad d_h = \frac{d_{\text{corr}}}{2}.$$

This iteration has to be done as long as one of the first cases is entered walking through the array. As clearly visible in Example 4, a flickering with inverse exponential decay can occur which ends after a specific threshold is met. In our implementation the floating point accuracy makes this threshold.

Example 4. Let $d_{\max} = 0.4$ and the individual be $(0.0, 0.4, 1.0)$. This means that the distance between two positions may not exceed 0.4 and the individual consists of two movements between 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 a distance not bigger than $d_{\max}$. 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 $d_{corr} = |0.6| - d_{\max} = 0.2$ is calculated and both positions are corrected half of the distance to the individual $(0.0, 0.5, 0.9)$. 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.

Begin	$(0.0, 0.4, 1.0)$
Iteration 1	$(0.0, 0.4, 1.0), i = 1, \lvert p_1 - p_2 \rvert \leq d_{\max}$
	$(0.0, 0.5, 0.9), i = 2, d_{\text{corr}} = 0.2$
Iteration 2	$(0.05, 0.45, 0.9), i = 1, d_{\text{corr}} = 0.1$
...	

□

3.3 Implementation

The application of the repair methods described above can be done in multiple ways. In the Lamarckian method, the repaired path is not only used for the representation but also written back to the population individual for further evolution. In the Baldwinian method, the repaired path is only used for the evaluation and is not written back to the population individual. For the *array of speeds* only the Baldwinian way can be utilized (in the following referenced as $\mathbf{Bal}_s$) as the repaired representation can not be written back to the population individual. For the *array of positions* both methods will be presented and evaluated in the results section referenced as $\mathbf{Lam}_1$ and $\mathbf{Bal}_1$ using the Lamarckian respectively the Baldwinian way together with the first repair method and $\mathbf{Lam}_2$, $\mathbf{Bal}_2$ for the second repair method. The source code of all the algorithms is based on the jMetal framework[1] and is made publicly available[2] (the data files used in this paper are also available on request).

[1] `http://jmetal.sourceforge.net/`

[2] `http://www.aifb.kit.edu/web/BlendingSystems`

4 Simulation Results

We test the algorithm by limiting the number of variables $l = 30$ (speed / position changes). Moreover, the maximum speed is set to the speed needed to place 20 layers of material with the Chevron stacking method. The starting position for the results of the *array of speeds* is fixed at position 0 (at the left side of the stockpile). The input data is described in Section 2 and we test the algorithms on four problems corresponding to four quality curves. (Other quality curves were tested and we got similar results.) As the quality varies over the time and can be classified once as high and once as low the obvious intuitive solution of the optimization is to spread the particles with high quality equally over the full stockpile width and do the same for the low quality particles. The result is a constant average cross section quality for the whole stockpile. This is a very simplified description of a non-trivial problem because the particles do not arrive sorted by quality so one could easily do the spreading but the quality varies quickly over the full range and the stacker can only be moved with a defined maximum speed.

Figure 5 illustrates two solutions of the optimization problem for a given quality curve. The quality curve which is the high frequency varying curve in the background can be clearly divided into two quality classes of high quality and low quality. One can easily see the spread of the high quality material first being mainly divided to the left side of the stockpile (at $y = 0$) and at the second bigger batch of high quality material mainly on the right side of the stockpile (at $y = 1$). Hence, the optimal solutions corroborate the intuitive rule mentioned in the last paragraph. This result is found by the evolutionary algorithm and is visible in almost all the optimal solutions. Figure 6 shows the sample run of the position based Baldwinian and Lamarckian techniques in ssNSGA-II algorithm for problem P1. We see that both the methods are able to find many well-spread solutions. Figure 6 also shows a knee region in the efficient front. The knowledge of knees is valuable to a designer [6, 8]. In order to statistically evaluate our results, we run each algorithm 45 times. Various attainment surface plots [9] are shown in Figure 7. From these we see that there might be a weakly efficient front corresponding to the minimizers of F_1 and F_2 (hence multiple solutions if we only minimize one objective using a single-objective algorithm).

For this real world problem, we do not know the exact location of the efficient front and hence, we use a hypervolume indicator [6] to compare the methods. Table 1 shows

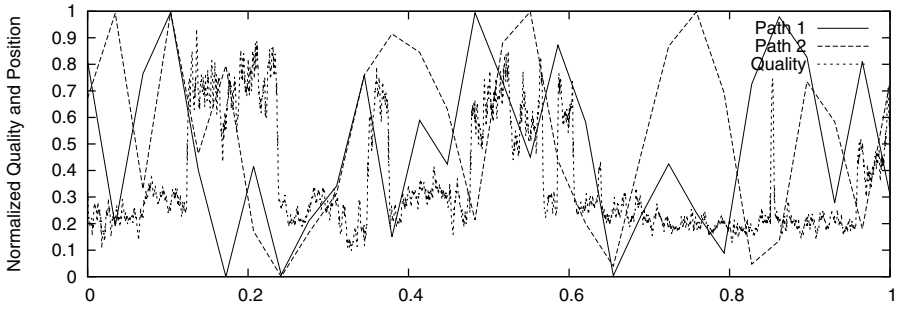

Fig. 5. Optimized solutions for the traverse path of a given input quality curve

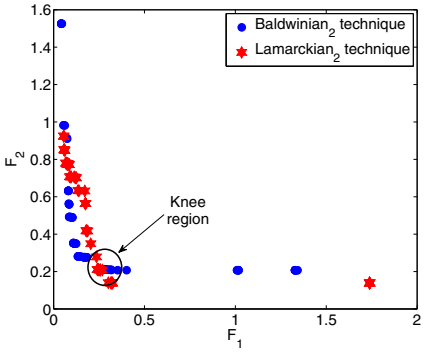

Fig. 6. Sample runs of ssNSGA-II with position based Baldwinian and Lamarckian methods for problem P1

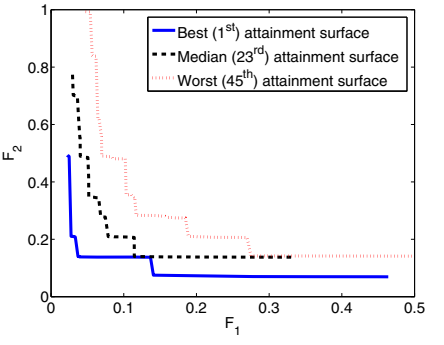

Fig. 7. Attainment surface plots of the ssNSGA-II algorithm with Lamarckian method for problem P1

Table 1. Hypervolume values for the four problems, corresponding to the reference point $(1, 1)^\top$. The hypervolume values for the Chevron method are an overestimate as we assume that the relative height difference is optimal (i.e., equal to 0). The values in dark and light grey correspond to the best and the second best algorithm (based on median values), respectively.

HV	$\mathbf{Bal}_s$	$\mathbf{Bal}_1$	$\mathbf{Bal}_2$	$\mathbf{Lam}_1$	$\mathbf{Lam}_2$
$HV_{Chevron,P1}$=0.85644					
$best_{P1}$	0.76844	0.84460	0.82855	0.81837	0.79758
$worst_{P1}$	0.60061	0.63456	0.67023	0.60957	0.59453
$median_{P1}$	0.66810	0.77881	0.76955	0.75582	0.72538
IQR_{P1}	0.07561	0.04862	0.05217	0.04637	0.06051
Chevron solution for P2 lies outside the region dominated by $(1, 1)^\top$					
$best_{P2}$	0.72217	0.82496	0.82939	0.85986	0.81691
$worst_{P2}$	0.59298	0.65282	0.62709	0.63912	0.56375
$median_{P2}$	0.65633	0.77066	0.74338	0.73368	0.68591
IQR_{P2}	0.05402	0.04700	0.06882	0.04597	0.09568
$HV_{Chevron,P3}$=0.69651					
$best_{P3}$	0.72205	0.82716	0.80311	0.80493	0.79385
$worst_{P3}$	0.57067	0.64642	0.64401	0.61633	0.59540
$median_{P3}$	0.62472	0.74523	0.73737	0.70537	0.71732
IQR_{P3}	0.04176	0.04757	0.05233	0.05825	0.06551
$HV_{Chevron,P4}$=0.16867					
$best_{P4}$	0.76140	0.88563	0.88149	0.86541	0.83725
$worst_{P4}$	0.65172	0.71243	0.73296	0.55602	0.55102
$median_{P4}$	0.71963	0.71253	0.73299	0.55602	0.55102
IQR_{P4}	0.04089	0.04142	0.03561	0.05149	0.05862

a hypervolume based statistical summary of the results. The reference point is chosen to be $(1, 1)^\top$ for simplicity, other values do not change the qualitative behavior. We see that the Baldwinian methods outperform the Lamarckian ones. Moreover, the *array of positions* representation usually delivered better results. The main reason for this is the higher stability of the individual, when changes are made to them by mutation or recombination. If an individual of the speed representation is modified at the begin of

the array then it affects the whole traverse path as the absolute reference position for the whole following path is shifted. This is not the case for the positions representation as changing one position only affects the traverse path in the immediate surrounding path to and from the position. For three out of four problems, we obtain a better hypervolume than the Chevron method, even if we assume that the Chevron is optimal in terms of the second objective. If we only consider the first objective, all the five algorithms produce better results than the Chevron method, for all the problems. The Chevron solution for P2 lies outside the region dominated by the reference point ($F_1 > 1.0$).

5 Conclusions and Future Work

The results presented in this paper show the optimization potential of bulk material blending beds that can be calculated and analyzed with the described methods using evolutionary algorithms. The methods presented in this paper provide a fast and flexible calculation environment with a scalable simulation system which can be adapted to various types of blending systems and parameters. We assumed in this paper that we have the full knowledge about input quality curve, however, the presented methods can be used to train and validate a Learning Classifier System for real time blending optimization. Moreover, techniques from [10, 8, 11] will be used to investigate the trade-off properties of the knee solutions in detail. Finally, attainment function [9] based statistical hypothesis and post-hoc tests shall also be conducted.

References

[1] Bond, J., Coursaux, R., Worthington, R.: Blending systems and control technologies for cement raw materials. IEEE Industry Applications Magazine, 49–59 (2000)
[2] Laurila, M.J., Bachmann, C.C.: X-ray fluorescence measuring system and methods for trace elements. US Patent US 2004/0240606 (2004)
[3] Kumral, M.: Bed blending design incorporating multiple regression modelling and genetic algorithms. International Journal of Surface Mining, Reclamation and Environment 17, 98–112 (2003)
[4] Pavloudakis, F., Agioutantis, Z.: Simulation of bulk solids blending in longitudinal stockpiles. Journal of the South African Institute of Mining and Metallurgy 106, 229–237 (2006)
[5] Fischer, A., Shukla, P.K.: A Levenberg-Marquardt algorithm for unconstrained multicriteria optimization. Oper. Res. Lett. 36(5), 643–646 (2008)
[6] Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley (2001)
[7] Durillo, J.J., Nebro, A.J., Luna, F., Alba, E.: On the Effect of the Steady-State Selection Scheme in Multi-Objective Genetic Algorithms. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 183–197. Springer, Heidelberg (2009)
[8] Deb, K., Gupta, S.: Understanding knee points in bicriteria problems and their implications as preferred solution principles. Engineering Optimization 43(11), 1175–1204 (2011)
[9] Fonseca, C.M., Guerreiro, A.P., López-Ibáñez, M., Paquete, L.: On the Computation of the Empirical Attainment Function. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 106–120. Springer, Heidelberg (2011)

[10] Shukla, P.K., Hirsch, C., Schmeck, H.: A Framework for Incorporating Trade-Off Information Using Multi-Objective Evolutionary Algorithms. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI, Part II. LNCS, vol. 6239, pp. 131–140. Springer, Heidelberg (2010)

[11] Shukla, P.K., Hirsch, C., Schmeck, H.: Towards a Deeper Understanding of Trade-offs Using Multi-objective Evolutionary Algorithms. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervós, J.J., Preuss, M., Richter, H., Silva, S., Simões, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Togelius, J., Urquhart, N., Uyar, A.Ş., Yannakakis, G.N. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 396–405. Springer, Heidelberg (2012)