WORKING PAPER
MANSHOLT GRADUATE SCHOOL

The semi-continous quadratic mixture design problem

Eligius M.T. Hendrix, L.G. Casado and I. García

DISCUSSION PAPER No. 24
2006

Mansholt Graduate School

Hollandseweg 1, 6706 KN Wageningen,
The Netherlands
Phone: +31 317 48 41 26
Fax: +31 317 48 47 63
Internet: http://www.mansholt.wur.nl/
e-mail: office.mansholt@wur.nl

Working Papers are interim reports on work of Mansholt Graduate School (MGS) and have received only limited reviews[1]. Each paper is refereed by one member of the Editorial Board and one member outside the board. Views or opinions expressed in them do not necessarily represent those of the Mansholt Graduate School.

The Mansholt Graduate School's researchers are based in three departments: 'Social Sciences', 'Environmental Sciences' and 'Agrotechnology and Food sciences' and two institutes: 'LEI, Agricultural Economics Research Institute' and 'Alterra, Research Institute for the Green World'. In total Mansholt Graduate School comprises about 250 researchers.

Mansholt Graduate School is specialised in social scientific analyses of the rural areas and the agri- and food chains. The Graduate School is known for its disciplinary and interdisciplinary work on theoretical and empirical issues concerning the transformation of agriculture, rural areas and chains towards multifunctionality and sustainability.

Comments on the Working Papers are welcome and should be addressed directly to the author(s).

[1] Working papers may have been submitted to other journals and have entered a journal's review process. Should the journal decide to publish the article the paper no longer will have the status of a Mansholt Working Paper and will be withdrawn from the Mansholt Graduate School's website. From then on a link will be made to the journal in question referring to the published work and its proper citation.

The semi-continous quadratic mixture design problem*

Eligius M.T. Hendrix, L.G. Casado and I. García
Operationele Research en Logistiek Groep, Wageningen Universiteit.
email: eligius.hendrix@wur.nl
Dpt. de Arquitectura de Computadores y Electrónica, Universidad de Almería.
email: leo@ace.ual.es, igarcia@ual.es

January 10, 2006

Abstract

The semi-continuous quadratic mixture design problem (SCQMDP) is described as a problem with linear, quadratic and semi-continuity constraints. Moreover, a linear cost objective and an integer valued objective are introduced. The research question is to deal with the SCQMD problem from a Branch-and-Bound perspective generating robust solutions. Therefore, an algorithm is outlined which is rigorous in the sense it identifies instances where decision makers tighten requirements such that no ϵ-robust solution exists. The algorithm is tested on several cases derived from industry.

keywords: Blending, Branch-and-Bound, Semi-continuity, Quadratic Programming.

1 Introduction

The mixture design problem consists of identifying mixture products, each represented by a vector $x \in R^n$, which meet certain requirements. The set of possible mixtures is mathematically defined by the unit simplex $S = \{x \in R^n | \sum_j x_j = 1.0;\ x_j \geq 0\}$, where the variables x_j represent the fraction of the components in a product x. In mixture design (blending) problems, the cost of the material, $f(x) = c^T x$ is minimised, where vector c gives the cost of the raw materials. In practical situations, such problems are solved on a daily base in fodder and petrochemical industry where often requirements are modelled by linear inequalities, see e.g. [10]. The current article is a result from a larger project on product design at Unilever Research. Products that are produced in

*This work has been partially supported by the Ministry of Education and Science of Spain through grants TIC2002-00228 and TIN2005-00447

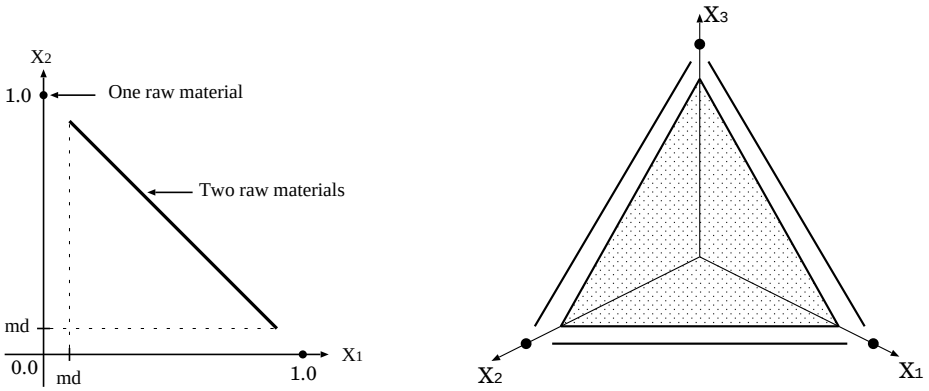

Figure 1: 2D and 3D simplices where the minimum dose region has been removed.

large quantities require extensive testing and careful designing where many aspects such as robustness, cost, choice and availability of raw materials etc. play a role. Four aspects get additional attention in the SCQMD Problem compared to the traditional blending problem [10].

1. The variables have a semi-continuous character.
2. The number of used raw materials is minimised in a separate objective function.
3. The requirements for the product are modelled by quadratic constraints.
4. A generated solution should have a certain robustness.

The semi-continuity of the variables models a minimum acceptable dose md that the practical problems reveal, i.e. either $x_j = 0$ or $x_j \geq md$. Figure 1 shows a graphical example of the search space in 2D (left hand side) and 3D (right hand side) consisting of unit simplices removing the space where the minimum dose constraint is not satisfied. The number of resulting sub-simplices (facets) is

$$\sum_{t=1}^{n} \binom{n}{t} = n! + 1, \tag{1}$$

where t denotes the number of raw materials in each sub-simplex.

A common way to formulate semi-continuity is as follows [10]:

$$x_j \leq \delta_j; \quad j = 1, \ldots, n \tag{2}$$

and

$$x_j \geq \delta_j \cdot md; \quad j = 1, \ldots, n \tag{3}$$

where implicitly the number of raw materials is minimised. So far, having linear constraints, a linear objective function and semi-continuous variables, a

general MILP (Mixed Integer Linear Programming) solution method can be applied. In practical problems, linear constraints $h_i(x) \leq;\ i = 1, \ldots, l$ exist that have the interpretation of bounding the design space of production. An additional feature is the appearance of quadratic inequalities that represent the requirements (production specifications) given as

$$g_i(x) = x^T A_i x + b_i^T x + d_i \leq 0; \quad i = 1, \ldots, m \tag{4}$$

where A_i is a symmetric n by n matrix, b_i is an n-vector and d_i is a scalar. The quadratic mixture design problem is studied in [5], where a specific Branch-and-Bound approach is constructed. The interesting characteristic is that even without the semi-continuity requirements we are dealing with a global optimization problem. One could also think of trying approaches based on quadratic inequalities by reformulating the semi-continuity by

$$x_j \cdot (md - x_j) \leq 0; \quad j = 1, \ldots, n \tag{5}$$

The question is whether this is the best thing to do. Probably it is better to make use of the semi-continuous character of the SCQMD problem and to construct a specific algorithm. Moreover, the practical problem requires some more additions to the model.

Not only the cost of the material, $f(x) = c^T x$ should be minimised, but also the number of raw materials in the mixture given by $\sum_{j=1}^{n} \delta_j$, where

$$\delta_j = \begin{cases} 1 & \textbf{if} \quad x_j > 0, \\ 0 & \textbf{if} \quad x_j = 0. \end{cases} \tag{6}$$

Discussion with the problem owners resulted into leaving out the idea of having fixed cost for the number of raw materials. However, it seemed more appropriate for the decision makers to take a multi-objective approach: The solution approach should provide the minimum possible cost for each. The idea is also sketched in Figure 2.

Applying standard software, the mathematical structure of the problem requires dealing with linear and quadratic (nonconvex) constraints and binary variables. From a complexity viewpoint finding solutions is a challenge, certainly when looking for rigorous approaches instead of applying heuristics.

The challenge gets bigger when considering another feature of the model. From a practical consideration it would be desirable, that small mistakes in the production process do not lead to "out of spec" products, i.e. the production based on the design should still fulfill the quadratic requirements despite small (ϵ) variations in the process. This can be handled by the concept of robustness. Consider the set of designs on the unit simplex fulfilling the quadratic requirements $D = \{x \in S | g_i(x) \leq 0; \quad i = 1, \ldots, m\}$. Robustness $R(x)$ of a design x with respect to D can be defined as $R(x) = \max_R$ s.t. $(x+h) \in D, \forall h, \|h\| \leq R$. Notice that for mixture problems $x + h$ is projected on the unit simplex. In [4] the robustness is calculated analytically for linear mixture design problems. However, this is not straightforward for quadratic inequalities. No analytic expression exists to determine the inequality nearest to x. Algorithms as described

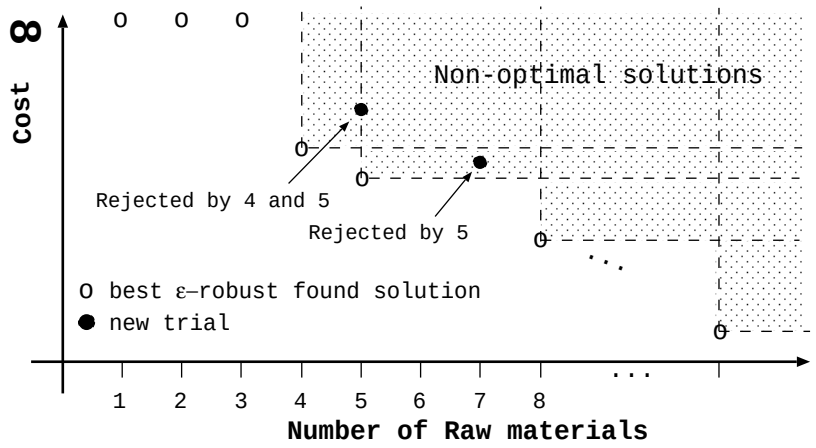

Figure 2: Rejection by domination. Pareto optimality

in [2] are needed. Moreover, [4] shows that determining the maximum robust point becomes a global optimization problem. We are mainly interested in generating ϵ-robust solutions, i.e. an element of $\{x \in D | (x+h) \in D, \forall h, \|h\| \leq \epsilon\}$. The research question is to compose an algorithm that has the aim to find an ϵ-robust solution as cheap as possible and that is rigorous in the sense it guarantees to identify the case that does not contain an ϵ-robust solution. Moreover, we have to keep track of the number of raw materials a design uses. Figure 2 shows how an ϵ-robust solution is discarded because it is dominated by another ϵ-robust solution with less cost and with a number of raw materials that is less or equal.

A specific algorithm has been designed for this case. The developed algorithm is based on the Branch-and-Bound concept applying simplicial partition sets. Several options are described and tested how to check feasibility of a subset and how to determine lower bounds on the robustness. In Section 2, a detailed description of the branch-and-bound algorithm to solve the above problem is given. Results obtained by solving test and real problems are given in Section 3 and conclusions and future work are discussed in Section 4.

2 Branch-and-Bound Algorithm

The scheme outlined here for solving the optimization problem falls into the general framework of Branch-and-Bound (B&B) algorithms. The basic idea in B&B methods consists of a recursive decomposition of the original problem into smaller disjoint subproblems until the solution is found. The method avoids visiting those subproblems which are known not to contain a solution. B&B methods can be characterized by four rules: *Branching, Selection, Bounding*, and *Elimination* [8, 9]. For problems where the solution is determined with a

desired accuracy, a *Termination rule* has to be incorporated.

Algorithm 1 : Branch-and-Bound algorithm for finding robust solutions of SCQMDP.

Funct B&B$(n, f, g_{1,\ldots,m}, h_{1,\ldots,l})$

1. Set $ns = n! + 1$ *number of simplices*
2. Set the working list $\Lambda := \{C_1, \ldots, C_{ns}\}$
3. Set the final list $Q := \{\}$
4. **while** ($\Lambda \neq \{\}$)
5. Select a simplex $C = C_k$ from Λ *Selection rule*
6. Evaluate C
7. Compute a lower bound $f^L(C)$ of f on C *Bounding rule*
8. **if** C cannot be eliminated *Elimination rule*
9. **if** C satisfies the termination criterion *Termination rule*
10. Store C in Q
11. **else**
12. Divide C into C_{ns+1}, C_{ns+2} *Division rule*
13. $ns = ns + 2$
14. $C = argmin\{f^L(C_{ns+1}), f^L(C_{ns+2})\}$ *Select the cheapest simplex*
15. Store $\{C_{ns+1}, C_{ns+2}\} \backslash C$ in Λ
16. Goto 6
17. **return** Q

We will denote a simplex by C_k, where k determines the order in which the simplex was generated and t_k specifies the number of raw materials of C_k. In the algorithm all the t_k vertices of a simplex C_k, denoted by $v_{k,j},\ j = 1, \ldots, t_k$, are evaluated, i.e. the values of the quadratic constraints $g_i(v_{k,j}),\ i = 1, \ldots, m$, the linear constraints $h_i(v_{k,j}),\ i = 1, \ldots, l$ and the cost value $f(v_{k,j})$ are determined. The cost value is used to update global upper bound values $f_t^U, t = 1, \ldots, n$. If $v_{k,j}$ happens to be feasible as well as ϵ-robust, it is stored as a Pareto optimal design if $f(v_{k,j}) \leq f_{t_k}^U$ and the upper bounds are updated accordingly.

Algorithm 1 starts by generating the initial set of $n! + 1$ sub-simplices (see Equation 1) which defines the search space resulting from removing the minimal dose region from the original simplex (see Figure 1). This initial set of simplices are stored in the working list Λ (line 2). While the working list is not empty, a simplex C_k from Λ is selected and evaluated (lines 5 and 6). If C_k can not be eliminated (line 8) and neither satisfies the termination criterion (line 9), it is bisected. From the two generated simplices the most expensive simplex is stored in the work list Λ while the algorithm proceeds with the cheapest one (Depth first search). Those simplices which satisfy the termination criterion are stored in the final list Q that determines the set of all simplices where the global ϵ-robust optimal mixture can be located (if any). The algorithm also stores the best ϵ-robust mixtures found for each number of raw materials $t = 1, \ldots, n$.

In the next subsections a detailed description of the rules of Algorithm 1 are

provided. The underlying theoretical foundation of the elimination procedures is given in other papers of the authors, see [1, 3].

2.1 Bounding Rule

The objective lower bound of the linear cost function f on C_k in the Bounding rule of Algorithm 1 can be taken as $f_k^L = \min_{j=1,\ldots,t_k} \{f(v_{k,j})\}$ due to the linearity of f. This lower bound is used by the Pareto-Cost elimination test as described in Subsection 2.5.4.

2.2 Branching Rule

The branching rule applied in Algorithm 1 consists of: given a simplex C_k, with $t_k \geq 2$, its longest edge is bisected generating two new simplices. Notice that this does not always generate a new point, as a generated vertex can be shared by different simplices. Due to bisection the length of the longest edge is at most twice the length of the shortest edge. Therefore simplices never get a needle shape [7]. If all edges are of equal length, the edge in between the cheapest and most expensive vertex is bisected. Therefore, the branching rule generates an on average more expensive and on average cheaper simplex. This helps the selection rule (Section 2.4) which gives higher priority to "cheaper" simplices.

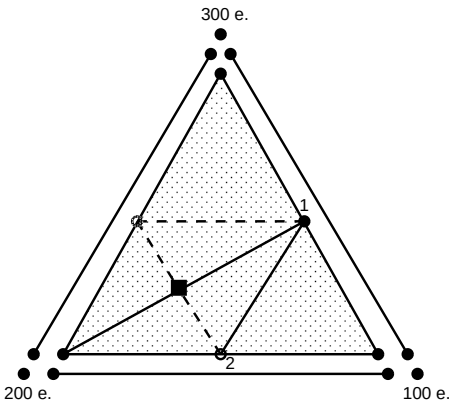

Figure 3: Example of division according to the raw material costs.

Figure 3 shows an example of the branching rule of Algorithm 1. First, the vertex labeled with 1 is generated taking into account the value of the cost at the vertices of the equilateral simplex. Vertex 2 is generated by bisecting the longest edge. Dashed lines represent future subdivisions that end in a vertex (drawn as a square) which is shared by four simplices. A discussion about lower and upper bounds on the number of simplices generated in the worst case by bisection can be found in [3].

2.3 Termination Rule

A simplex is not divided further when its size is smaller than the value of the accuracy α. The size is given by the length of its longest edge, i.e. $Size(C_k) = \max_{v,w \in C_k} \|v - w\|$. If it has not been rejected by the elimination rules, it is saved in a final list Q. Algorithm 1 ends when the working list Λ is empty. List Q provides the set of simplices where the optimal ϵ-robust solution can be found (if any).

2.4 Selection Rule

The selection rule has been designed to reach two goals: to facilitate discarding simplices with a large number of raw materials and to reduce the memory requirements. The first goal is met by giving priority to simplices with a few number of raw materials and low cost. Simplices with more raw materials and/or higher cost value can be dominated by those with less raw materials, which improves the efficiency of the Pareto test (see Subsection 2.5.4). In the algorithm, the cost of a simplex is measured by the sum of the costs at its vertices, i.e. $Cost(C_k) = \sum_{j=1}^{t_k} f(v_{k,j})$.

The second goal is met by applying a depth-first selection rule. Once a simplex is selected and divided, its cheaper child will be the new selected simplex until no further subdivision is allowed. This reduces the memory requirement of the algorithm.

2.5 Rejection Rule

The rejection rule consists of a set of tests which takes into account feasibility conditions associated to the linear and quadratic constraints and the requirements related to the robustness of the solution as well as the minimization of the number of raw material and the cost function. The following rejection tests have been designed to be applied to the selected simplex. The order in which these tests are applied does not affect the final result of the algorithm in terms of the final solution but it may influence the efficiency related to the computational effort (time).

2.5.1 Linear infeasibility.

If for one of the linear constraints $h_i(x) \leq 0$ all vertices are infeasible ($h_i(v_{k,j}) > 0, j = 1, \dots t_k$), then C_k is discarded because C_k does not fulfill $h_i(x) \leq 0$.

2.5.2 Quadratic infeasibility.

Given a simplex C_k and the set of quadratic constraints $g_i(x) \leq 0, i = 1, \dots, m$, C_k can be rejected if $\forall x \in C_k \ \exists \ i \ g_i(x) > 0$. This is equivalent to $\forall x \in C_k \ \max_i g_i(x) > 0$; i.e. $\min_x \max_i g_i(x) > 0$.

In [1], around each vertex $v_{k,j}$, a so-called infeasibility sphere $B_{k,j}$ is defined that cannot contain a feasible point

$$B_{k,j} = \{x \in R^n, \|x - v_{k,j}\|^2 < \rho_{k,j}^2\} \tag{7}$$

Two possible ways of calculating a value for $\rho_{k,j}$ are described in [1]. One way is illustrated in Figure 4. Based on Lipschitz constant:

$$L_i(C_k) = \max_{v \in C_k} \|\nabla g_i(v) - \frac{\mathbf{1}^T \nabla g_i(v) \mathbf{1}}{n}\| \tag{8}$$

one can derive an infeasibility radius

$$\rho_{k,j} = \max_i \frac{g_i(v_{k,j})}{L_i(C_k)} \tag{9}$$

Figure 4 sketches (9) for a 2-dim case with two vertices and two requirements

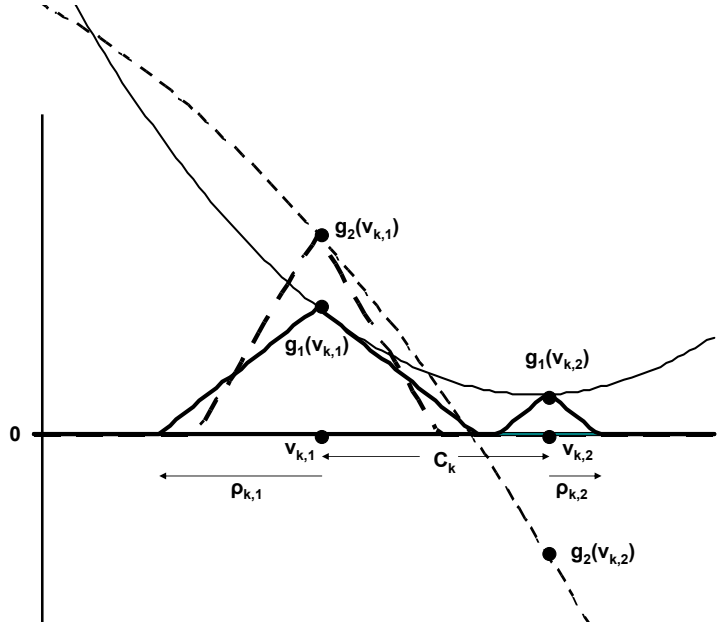

Figure 4: Example of quadratic infeasible radius taking $L_i = max_{x \in C_k} \mid g_i'(x) \mid$

$g_i(x) \leq 0$. Although it is known that the vertices are infeasible, the two spheres do not cover the simplex completely.

A second way to derive infeasibility spheres is to make use of the minimum (negative) eigenvalue η_i of matrix A_i. In [3] it is proven that a valid radius is given by

$$\rho_{k,j}(C) = \max_{i \ with \ \eta_i < 0} \frac{\|\nabla g_i(v_{k,j})\| - \sqrt{\|\nabla g_i(v_{k,j})\|^2 - 4\eta_i g_i(v_{k,j})}}{2\eta_i} \tag{10}$$

which appeared to be useful (larger) for small simplices. If none of the vertices of C_k happens to be feasible, the spheres can be used to prove that C_k cannot contain a feasible quadratic solution. For instance, $\rho_{k,j} > \max_{x \in C_k} \|v_{k,j} - x\|$ means that C is completely covered by $B_{k,j}$. The following tests to prove infeasibility of simplex C_k have been used by the algorithm.

SCTest (Single Cover test): One of the spheres $\rho_{k,j};\ j = 1, \ldots, t_k$ covers the simplex completely; i.e. C_k is proven infeasible if there exists a vertex $v_{k,j}$ such that $\rho_{k,j} > \max_s ||v_{k,j} - v_{k,s}||\ s = 1, \ldots t_k$.

MCTest (Multiple Cover Test): A simplex C_k which is not covered by a single sphere $B_{k,j}$ can still be covered by $\cup_{j=1}^{t_k} B_{k,j}$. In [3], is proven that if there exists a mixture $x \in C_k$ which is covered by all spheres i.e. $x \in \cap_{j=1}^{t_k} B_{k,j}$ then $C_k \subset \cup_{j=1}^{t_k} B_{k,j}$. This means that all the possible mixtures in C_k are covered by at least one sphere $B_{k,j}$ and consequently C_k is infeasible. The two heuristic points suggested in [1] are considered good candidate mixtures to be covered by all the spheres. The mixture described by weighted average (11) is considered a good guess.

$$\xi_k = \frac{1}{\sum_j \frac{1}{\rho_{k,j}}} \sum_j \frac{v_{k,j}}{\rho_{k,j}} \tag{11}$$

In case point ξ_k is not covered by the smallest sphere, the candidate will be $\theta_k = v_{k,s} + (\frac{\rho_{k,s}}{\|\xi_k - v_{k,s}\|} - \delta)(\xi_k - v_{k,s})$, with $s = argmin_j \rho_{k,j}$ which is a point just within the interior of the smallest sphere.

PCTest : If the SCTest and MCTest fail, Algorithm 1 focuses on the unfeasibility sphere centered at the generated interior point, either ξ_k or θ_k. It is known to be infeasible for at least one of the constraints. The quadratic constraints g_i and the corresponding radius of a new infeasibility sphere ρ are generated. The advantage of using an interior point is that its distance to the farest vertex is smaller than for the largest distance in between the vertices.

2.5.3 ϵ-infeasibility.

The ϵ-robustness requirement gives the opportunity to reject a simplix C_k that is in total so close to an infeasible solution, that it certainly cannot contain an ϵ-robust solution. A simplex C_k with an infeasible point x such that $\max_j \|x - v_{k,j}\| < \epsilon$, can not contain an ϵ-robust solution. The ϵ-infeasibility test checks this condition for $x = v_{k,j}$ $(j = 1, \ldots, t_k)$, $x = \xi_k$ and possibly θ_k.

2.5.4 Pareto bounding.

We are dealing with the minimization of the number of raw materials and the composition cost. If an ϵ-robust solution has been found with t_k raw materials and a lower cost than f_k^L, then C_k can be eliminated, i.e. discard C_k if $f_{t_k}^U < f_k^L$.

The algorithm keeps track of f_t^U which is initially set to infinity. When an ϵ-robust mixture $v_{k,j}$ is found with $f(v_{k,j}) < f_{t_k}^U$, the value of f_t^U is updated for $t = t_k, \ldots, n$. Algorithm 1 returns the Pareto vector f^U and the corresponding mixtures.

2.6 Robustness underestimation

Algorithm 1 has been designed to deal with mixture design problems where the solution (if any) has to be ϵ-robust. Robustness $R(x)$ of a design x with respect to D has been defined as $R(x) = \max_R$ s.t. $(x + h) \in D, \forall h, \|h\| \leq R$. For quadratic inequalities this is not easy to determine. The algorithms applies an underestimation $R^L(x)$ based on either a global Lipschitz constant, or maximum positive eigenvalue similar to (9) and (10). For the details we refer to [1]. In this way the algorithm identifies an evaluated point x as being ϵ-robust if $R^L(x) > \epsilon$.

3 Numerical testing of the algorithm

In this section numerical results of the evaluation of Algorithm 1 are discussed. The wider question is whether the algorithm is able to solve realistic problems in reasonable time. Earlier results have shown the functioning of the robustness calculation and infeasiblity tests for small examples, see [1]. The further question is to add the realistic perspective of semi-continuity and Pareto optimality of the problem to be solved. Moreover, results were given for two cases provided by Unilever Research. The exact numbers were mutated, but "could be realistic". Solutions found by standard solvers for the cases without robustness considerations and semi-continuity were provided. As far as we know, no other optimization research has tried to generate robust solutions to such problems in a rigorous way. The question is whether the algorithm is able to do so for these practical cases.

Executions were carried out on a dual Intel XEON 3.6GHz with 12GBytes of RAM running Linux. The algorithm was coded in C and uses the Lapack library. To check the algorithm two test problems taken from [5] were tested. Both are three dimensional problems. The first one (RumCoke) has two linear constraints and two quadratic constraints. The second one (Case 2) was taken from an industrial example having five quadratic constraints. Additionally, the algorithm was tested with the two seven dimensional problems (UniSpec1 and UniSpec5b) provided by Unilever Research based on similar quadratic functions, but with different requirements. UniSpec1 has one linear constraint and five quadratic constraints and UniSpec5b has four quadratic constraints. A detailed description of these four mixture problems is given in the Appendix. For all problems the robustness is given by $\epsilon = \frac{\sqrt{2}}{100}$ and the semi-continuity by a minimum dose value of $md = 0.03$.

It is worth mentioning that in addition to the ϵ-robust solution of the blending problem (if any), Algorithm 1 is able to provide a set of simplices which contains all the feasible solutions that may also be ϵ-robust mixtures. Our nu-

merical results show that it is possible to provide solutions for hard problems in less than half an hour using.

Table 1 shows the figures obtained by the algorithm. All problems were solved with an accuracy of $\alpha = \epsilon$. The meaning of the notation is:

Problem	Problem name.
NSimplex	Number of evaluated simplices.
NVertex	Number of evaluated vertices.
End NSimplex	Number of simplices in list Q.
End NVertex	Number of vertices associated to simplices in list Q.
ϵ-Infeas.	Number of simplices rejected by ϵ-infeasibility test applied to the vertices.
Pareto	Number of simplices rejected by Pareto bounding.
SCTest	Number of simplices rejected by SCTest.
MCTest	Number of simplices rejected by MCTest.
ξ-θ-ϵ-infeas.	Number of simplices rejected by the ϵ-infeasibility test applied to ξ or θ.
PCTest	Number of simplices rejected by PCTest.
LC	Number of simplices rejected by Linear infeasibility.
Time	The running time in hh:mm:ss format.
Memory	The memory required by the algorithm.
N. Sol.	A binary vector which shows if for a given number of raw materials the algorithm found a solution.

For UniSpec1 and UniSpec5b the solutions found by Algorithm 1 improves or are similar to previous known solutions, whereas they are guaranteed to be ϵ-robust. Theoretically the algorithm converges in a "finite number of steps", which practically still can require more than a human life time. To deal with the more than 100 million of simplices, efficient data structures have been designed and efficient data handling is necessary.

The best ϵ-robust Pareto solutions found were:

RumCoke : No robust solution was found.

Case2 : $f(0.552539, 0.293046, 0.154414) = 1.414801$

UniSpec1 : $f(0.654218, 0.345781, 0.0, 0.0, 0.0, 0.0, 0.0) = 114.345781$
and $f(0.428125, 0.0, 0.435234, 0.0, 0.136640, 0.0, 0.0) = 111.09$

UniSpec5b : $f(0.165078, 0.0, 0.303711, 0.531211, 0.0, 0.0, 0.0) = 118.779766$
and $f(0.129687, 0.0, 0.215625, 0.274063, 0.380625, 0.0, 0.0) = 116.434062$.

The solutions for the case UniSpec5b means that the cheapest solution has 4 raw materials, which was also confirmed by solutions found before. The case was designed in such a way that it is much harder to find feasible solutions than for the UniSpec1 case, which has a solution with even two raw materials. As soon as ϵ-robust mixtures are found, the Pareto bounding is able to discard

Table 1: Numerical results

Problem	RumCoke	Case2	UniSpec1	UniSpec5b
NSimplex	581	401	73,831	97,183,929
NVertex	257	184	34,066	33,706,308
End NSimplex	51	28	701	1,763,137
End NVertex	44	26	295	136,820
ϵ-infeas.	132	86	4,566	14,737,579
Pareto	0	9	6,343	2,551,778
SCTest	76	61	11,899	16,155,990
MCtest	33	19	5,686	10,396,930
ξ-θ-ϵ-infeas.	0	1	431	2,457,144
PCTest	0	0	270	529,470
LC	2	0	7,083	0
Time	00:00:00	00:00:00	00:00:00	00:17:57
Memory	17 KB	12 KB	294 KB	635 MB
N.Sol.	0,0,0	0,0,1	0,1,1,0,0,0,0	0,0,1,1,0,0,0

large portions of the search region. On the other hand, the ϵ-infeasible test is only efficient when simplices are small enough, i.e. deep enough in the search tree. One should keep in mind that the order in which the rejection tests are applied is the order in Table 1. Therefore, one cannot determine the efficiency of each test individually because it may depend on the order in which they are applied.

4 Conclusions and Future work

The semi-continous quadratic mixture design problem is described as a (Vector) optimization problem that adds to the classical blending problem the following features. 1. The variables have a semi-continuous character. 2. The number of used raw materials is minimised in a separate objective function. 3. The requirements for the product are modelled by quadratic constraints.

Such a problem could be approached with an algorithm that handles (non-convex) quadratic constraints and binary variables. A specific algorithm is described with the aim to generate solutions that have a certain robustness. The algorithm is rigorous in detecting cases that do not contain ϵ-robust solutions. The new aspect of the algorithm is that it guarantees that the generated solutions are ϵ-robust. The termination rule based on parameter α guarantees convergence in a finite number of steps, see e.g. [6]. Practically, efficient data handling is necessary to handle the 100 million of generated subsets within half an hour. Keeping in mind the exponential behaviour in dimension of the worst case number of subsets, one is lucky the current dimension represents the size the industrial counterpart is interested in.

Sufficient challenges for further investigation remain. As noticed, the effi-

ciency depends on the instance. Further tests with real cases, not reported due to confidentiality of data, gave promising results in terms of the amount of calculation time and memory requirement. The next step is to extend the model in the direction of multiple end products, in literature called the "multi-blend" problem. In this problem the individual product simplices are connected due to common constraints on availability and use of raw materials.

References

[1] Leocadio G. Casado, Eligius M. T. Hendrix, and Inmaculada García. Infeasibility spheres for finding robust solutions of blending problems with quadratic constraints. *Journal of Global Optimization*, 2006. submitted to Journal of Global Optimization.

[2] Eligius M. T. Hendrix. *Global optimization at work*. Wageningen University, Wageningen, 1998.

[3] Eligius M. T. Hendrix, L. G. Casado, and I. García. On blending with Lipschitzian requirements. Technical Report 16, Mansholt Graduate School, http://www.sls.wau.nl/mi/mgs/publications, 2005. submitted to Mathematical Programming.

[4] Eligius M. T. Hendrix, Carmen J. Mecking, and Theo H. B. Hendriks. Finding robust solutions for product design problems. *European Journal of Operational Research*, 92:28–36, 1996.

[5] Eligius M. T. Hendrix and Janos D. Pintér. An application of Lipschitzian global optimization to product design. *Journal of Global Optimization*, 1:389–401, 1991.

[6] R. Horst, P.M. Pardalos, and N.V. Thoai. *Introduction to Global Optimization*, volume 3 of *Noncovex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, Holland, 1995.

[7] Reiner Horst. On generalized bisection of n-simplices. *Mathematics of Computation*, 66(218):691–698, 1997.

[8] T. Ibaraki. Theoretical comparisons of search strategies in branch and bound algorithms. *Int. J. Comput. Inform. Sci.*, 5:315–344, 1976.

[9] L. G. Mitten. Branch and bound methods: general formulation and properties. *Operations Research*, 18:24–34, 1970.

[10] H. P. Williams. *Model Building in Mathematical Programming*. Wiley & Sons, Chichester, 1993.

Appendix; Test Problems

RumCoke

Dimension = 3; Raw material cost = $(0.1, 0.7, 4.0)$
Linear Contraints:
$h_1(x) = -1.5x_1 + 0.5x_2 - 0.5x_3 \geq 0.0$
$h_2(x) = 0.3x_1 - 0.5x_2 - 0.3x_3 \geq 0.0$
Quadratic constraints ($g_i(x) = x^T A_i x + b_i^T x + d_i \leq 0; \quad i = 1, 2$):
$A_1[3 \times 3] = (0, -16, 0, -16, 0, 0, 0, 0, 0)$
$b_1[3 \times 1] = (8, 8, 0)$; $d_1 = -1$
$A_2[3 \times 3] = (10, 0, 2, 0, 0, 0, 2, 0, 2)$
$b_2[3 \times 1] = (-12, 0, -4)$; $d_2 = 3.7$

Case2

Dimension = 3; Raw material cost = (1.1, 1.7, 2.0)
Quadratic constraints ($g_i(x) = x^T A_i x + b_i^T x + d_i \leq 0; \quad i = 1, \ldots, 5$):
$A_1[3{\times}3] = (0.001, -0.001, 0.0085, -0.001, 0.008, -0.0105, 0.0085, -0.0105, -0.021)$
$b_1[3 \times 1] = (-0.0145, -0.0205, 0.073)$; $d_1 = -0.0165$
$A_2[3 \times 3] = (-0.004, 0.0005, 0.002, 0.0005, -0.001, -0.003, 0.002, -0.003, 0.014)$
$b_2[3 \times 1] = (0.0155, 0.0515, -0.121)$; $d_2 = -0.006$
$A_3[3{\times}3] = (20.605, -5.087, -10.9885, -5.087, 32.003, -43.476, -10.9885, -43.476, -81.278)$
$b_3[3 \times 1] = (0.1995, -0.097, 126.7685)$; $d_3 = -20.5063$
$A_4[3{\times}3] = (0.766, -0.1205, 2.4735, -0.1205, 0.528, 1.9835, 2.4735, 1.9835, -7.822)$
$b_4[3 \times 1] = (-2.432, -15.191, 10.712)$; $d_4 = 3.21125$
$A_5[3{\times}3] = (116.75, -3.09, 168.553, -3.09, -67.424, 515.114, 168.553, 515.114, -845.215)$
$b_5[3 \times 1] = (-287.43, -645.926, 354.537)$; $d_5 = 115.0953$

UniSpec1

Dimension = 7; Raw material cost = (114, 115, 107, 127, 115, 106, 108)
Linear Contraint:
$h_1(x) = 0.1493x_1 + 0.6927x_2 + 0.4643x_3 + 0.7975x_4 + 0.5967x_5 + 0.6235x_6 + 0.5284x_7 \geq 0.35$
Quadratic constraints ($g_i(x) = x^T A_i x + b_i^T x + d_i \leq 0; \quad i = 1, 2, 3$):
$A_1[7 \times 7]$ = (-1.473, 8.215, -27.204, 46.119, 2.059, -11.929, -12.768, 8.215, 37.733346, 5.127, 95.691, 34.954, 20.165, 19.445, -27.204, 5.127, -21.743, 36.843, -7.126, 4.029, -4.152, 46.119, 95.691, 36.843, 189.643, 93.359, 52.904, 54.802, 2.059, 34.954, -7.126, 93.356, 31.885, 7.528, 10.248, -11.929, 20.165, 4.029, 52.904, 7.528, 11.951, 10.964, -12.768, 19.445, -4.152, 54.802, 10.248, 10.964, 7.197)
$b_1[7 \times 1]$ =(4.5675, 34.7289, 70.5707, -82.2761, 29.3169, 71.0818, 63.7614); d_1 =-35

$A_2[7 \times 7]$ =(1.35, -4.41, 17.60, -92.45, 2.74, -29.94, -14.05, -4.41, -39.13, -6.11, -126.38, -29.81, -63.42, -43.97 , 17.60, -6.11, 15.45, -76.60, 5.93, -44.05, -20.54 , -92.45, -126.38, -76.60, -240.64, -117.46, -125.18, -114.98, 2.74, -29.81, 5.93, -117.46, -22.90, -47.37, -30.68, -29.94, -63.42, -44.05 , -125.18, -47.37, -73.39, -73.99, -14.05, -43.97, -20.54, -114.98, -30.68, -73.99, -55.33)
$b_2[7 \times 1]$ =(-2.1232, -9.0403, -42.2072, 190.5292, -9.9529, 1.8162, 5.1622); d_2 =10
$A_3[7 \times 7]$ =(-0.670, 4.284, -12.837,23.708, 1.677, -8.964, -4.859, 4.284, 21.380, -1.188, 28.990, 13.216, 17.177, 16.620, -12.837, -1.189, -21.376, 9.841, -7.298, -10.043, -8.981, 23.708, 28.990, 9.841, 49.385, 25.574, 15.561, 21.666, 1.677, 13.216, -7.298, 25.574, 8.419, 4.149, 6.595, -8.965, 17.177, -10.043, 15.561, 4.149, 1.090, 6.292, -4.859, 16.620, -8.981, 21.666, 6.594, 6.292, 5.906)
$b_3[7 \times 1]$ =(0.7097, -13.0982, 27.5078, -49.1608, -7.3725, 33.6731, 11.3136); $d_3 = -2$

UniSpec5b

Dimension = 7; Same raw material cost and similar quadratic requirements as UniSpec1
Quadratic constraints ($g_i(x) = x^T A_i x + b_i^T x + d_i \leq 0; \quad i = 4, 5, 6, 7$):
$A_4 = -A_1$; $b_4 = -B_1$; $d_4 = 45$
$A_5 = -A_2$; $b_5 = -B_2$; $d_5 = -21$
$A_6[7 \times 7]$ =(0.0, -11.556, -1.114, 14.690, -11.411, 0.121, -0.150, -11.556, -3.316, -2.116, 7.313, -8.800, 19.897, 9.051, -1.114, -2.116, 4.728, 16.250, -4.535, 18.319, 11.537, 14.690, 7.313, 16.250, 40.428, 9.766, 21.512, 15.266, -11.412, -8.800, -4.535, 9.766, -10.165, 10.088, 1.889, 0.121,19.897, 18.319, 21.511, 10.088, 28.569, 27.239, -0.150,9.051, 11.537, 15.266, 1.889, 27.239, 19.965)
$b_6[7 \times 1]$ =(1.7278, 23.5166, 5.6724, -32.0798, 19.0154, 16.5074, 7.31003); $d_6 = -5$
$A_7 = A_3$; $b_7 = B_3$; $d_7 = -1$