T-2615

INGOT BLENDING WITH DISCRETE INPUTS

by

Susan E. Melvin

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

T-2615

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mineral Economics).

Golden, Colorado

Date April 15, 1982

Signed: Susan E. Melvin
Susan E. Melvin

Approved: Robert E. D. Woolsey
Robert E. D. Woolsey
Thesis Advisor

Golden, Colorado

Date 4/16/82

C. W. Berry
Dr. Charles W. Berry, Head
Department of Mineral Economics

Abstract

This thesis discusses three approaches to the problems encountered when an alloy blend must be made with discreet, non-continuous inputs. A case study taken from industry is used to describe the problem parameters; composite data from this company is expanded to create a working data set from which the problem formulation is developed.

The first approach to this problem involves the use of the partial enumeration algorithm of Lawler and Bell for 0-1 problems. This algorithm will produce the optimal solution as well as feasible solutions up to that point. Computer run time for this optimization algorithm increase with the number of variables to be considered and in the 60 variable case considered here, runs into hours. It is questionable whether or not a search for the optimal blend can be justified in terms of both computer expense and carrying costs for unused inventory.

The second approach addresses this problem in terms of ordinal scales of vectors that break into a feasible region. This method is similar to that of Senja and Toyoda but does not employ an objective function, therefore the solutions it produces are not optimal but each one is feasible. Computer run time for this algorithm is about twenty seconds for the same data set used for the partial

enumeration technique.

The third method modifies the heuristic of Riesco and Thomas to produce a non-computerized quick and dirty routine that will give a set of feasible solutions from which to chose. While impractical for problems involving many blends per day, it does have application when the blends required are five or fewer.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF PLATES

ACKNOWLEDGEMENTS

"Cooperate and graduate" was the phrase I heard so many times when I first started this program that I thought it was a mantra used for cosmic enlightment. I found, though, that it was the truth and now must thank students, staff, and faculty who did indeed cooperate so fully with me.

Special thanks go to a special few:

To Drs. Ruth Maurer, Chuck Lienert, and Gene Woolsey- committee members, teachers, and mentors - who have cajoled me through this project as well as graduate school;

To Bob White, who made computers seem as helpful as he was;

To Doran Greening, who could always make me see the humor in any situation;

To Jean Goldberg, and her inexhaustible supply of tea and empathy;

And of course to Doug, my husband and best friend too.

INTRODUCTION

Blending requirements for the vast majority of metallurgical processes are determined by systems of simultaneous linear equations. These can be simplified into charge flow calculation sheets or can have the more elegant look of a linear program that uses the simplex technique. In either case, limited resources are allocated among competing activities to obtain the best (most profit/least cost) solution (Hillier and Lieberman, 1980).

This is quite satisfactory for many blending problems. The input variables are continuous and measured by weight or volume and valued by percent weight of the various components of the crushed ore. The result is a molten solution that could be used as an input variable for further refinement to the blend or could be molded into the final form.

A problem arises, however, when discrete inputs, not continuous ones, must be used. This occurs in some cases when the first blend produces a first-stage unit that will be used in a unit form for further blending. For example, bars of silver bullion are made with various assay values and tramp elements. If these bars are used to make silver coins or jewelry, a secondary-stage blending is needed to produce a product that has a certain assay value and

meets the bounds on other constituent elements.

Metallurgical literature on discrete input blending is silent on the means to treat problems of this type, but this problem is well addressed in integer programming literature, most specifically for 0-1 problems.

The discrete input blending problem confronted in the remainder of this thesis is a problem taken from industry. The company will be called RF Enterprises (a fictitious name). The specifications for input variables and the resultant output is propriatery information and is given here in disguised form as composite data from two sources. This data will be expanded into a discrete variable set having weights, assays, and constituent element values for both the input and output variables.

The problem statement for RF Enterprises will be given in the next section, to be followed by data expansion and problem formulation. After this will be a section detailing three approaches to the solution of this problem with corresponding results. The last section will contain conclusions.

PROBLEM STATEMENT

RF Enterprises produces two qualities of iron ingots from iron pigs. The iron pigs, received as inventory from two sources, have a stated assay value, a known weight, and 16 other constituent or tramp elements. All of these components can vary for each pig. RF Enterprises receives a statement from their suppliers giving the specifications for each pig but has no immediate control over the shipment. The companies that produce these iron pigs do so in a primary blend process that meets minimally acceptable standards for pigs. While it is possible for them to blend and process for a very high assay value, the supplying companies do not do this for it would mean much higher production costs for them that they could not easily pass along.

The problem RF Enterprises faces is to blend these pigs into ingots of Alloy A and Alloy B having assay values of 95.56-96.96 percent and 96.96-100 percent, respectively. The total ingot weight cannot exceed a given upper bound or the ingot is impractically large, nor can it be less than a given lower bound or production runs are too costly. The individual tramp elements when combined with the same elements from other pigs cannot exceed a certain percent weight of the total ingot. The lower bounds for all these

tramp elements is zero. RF Enterprises would like to produce as many ingots as possible for both alloys and would like to use all of their inventory stock if possible.

RF Enterprises has been using an explicit enumeration technique that examines between five and six million possible combinations of iron pigs in order to determine which should be blended to make ingots. This is obviously a very time consuming and costly computer process and for these reasons has proved unsatisfactory.

Considerable improvement can be achieved over the method of explicit enumeration, as will be shown in subsequent sections of the thesis. The next two sections will deal with data expansion and problem formulation. This data set will be used in a partial enumeration algorithm, a vector analysis algorithm, and a quick and dirty routine.

DATA EXPANSION

Composite data for the iron pigs supplied to RF Enterprise is given in Tables 1 and 2. This represents a Gaussian distribution for all 16 tramp elements, assay values, and weights of iron pigs from both sources. The mean values for all constituent elements, assays and weights are normalized to one in these data. The upper and lower bounds on all constituent elements from either source are actually equal so an adjustment is made to the source 2 data to reflect this as seen in Table 3. This is done by dividing Source 1 upper bound entries by the corresponding Source 2 entries and using the resultant value to multiply all entries in a constituent row of Source 2.

Some things are striking about these data. First, the tramp elements, except for #14 in Source 1 do not appear to be constraining since they can range beyond five standard deviations. The important and most binding constraints are weight and assay value of an ingot. Secondly, the minimally acceptable assay of an ingot produced from Source 1 is higher than the average pig from this source. One might suspect that many of these pigs will prove unacceptable for any blend.

Random numbers having a mean of zero and a standard deviation of one are generated for each constituent for a sample population of sixty pigs, thirty from each source. Percentage values for the seventeen chemistries and the weight of each pig in the population are determined by multiplying a random number by the S value for the constituent and adding to this the X value for that same constituent. The values for chemistries and weights of all pigs are used in combination with the upper and lower count on ingot specifications, to produce an inventory of pigs. This is further described in the Problem Formulation section.

TABLE 1

SOURCE 1 DATA

Constituents	$\bar{X}$	S	U	L
1	1	.42	2.02	0
2	1	.87	5.72	0
3	1	.26	2.45	0
4	1	.08	3.57	0
5	1	.31	3.02	0
6	1	.89	11.33	0
7	1	1.03	2.49	0
8	1	.45	2.05	0
9	1	.56	2.66	0
10	1	0	10.0	0
11	1	.86	2.60	0
12	1	1.07	2.79	0
13	1	0	20.0	0
14	1	1.71	1.10	0
15	1	0	30.0	0
16	1	0	17.30	0
17(assay)	1	.003	1.0356	1.0014
18(weight)	1	.08	4.92	3.87

TABLE 2

SOURCE 2 DATA

Constituents	$\bar{X}$	S	U	L
1	1	.42	1.61	0
2	1	.26	19.57	0
3	1	1.29	45.07	0
4	1	.28	90.42	0
5	1	1.1	14.78	0
6	1	.28	18.12	0
7	1	.15	19.01	0
8	1	.28	2.52	0
9	1	0	0	0
10	1	.88	17.53	0
11	1	.44	5.21	0
12	1	0	20.0	0
13	1	.27	21.83	0
14	1	0	20.0	0
15	1	1.41	30.8	0
16	1	.01	16.78	0
17(assay)	1	.001	1.02	.9871
18(weight)	1	.02	4.08	3.21

TABLE 3

ADJUSTED SOURCE 2 DATA

Constituents	$\bar{X}$	S	U	L
1	1.25	.527	2.02	0
2	.29	.08	5.72	0
3	.05	.06	2.45	0
4	.04	.01	3.57	0
5	.20	.22	3.02	0
6	.63	.18	11.33	0
7	.13	.02	2.49	0
8	.81	.23	2.05	0
9	.44	0	2.66	0
10	.57	.50	10.0	0
11	.50	.22	2.6	0
12	.14	0	2.79	0
13	.92	.25	20.0	0
14	.06	0	1.10	0
15	.97	1.37	30.0	0
16	1.03	.01	17.3	0
17(assay)	1.0153	.001	1.0356	1.0014
18(weight)	1.21	.02	4.92	3.87

PROBLEM FORMULATION

The form of the equations for upper and lower weight percents for constituent elements for iron ingots, total weight, and the objective function are developed in this section. To establish a common vocabulary, let:

a_{ij} = percent of constituent j in pig i (j=1...m)

w_i = weight of pig i (i=1...n)

x_i = 0,1 (a switch variable indicating non-use or use)

U_j = upper bound on percent composition in ingot of element j

L_j = lower bound percent on composition in ingot of element j

W_u = upper weight limit on an ingot

W_l = lower weight limit on an ingot

The general form for upper and lower weight percentages for elements, j, in an ingot is given by:

$$L_j < \frac{\sum_{i=1}^{n} w_i a_{ij} x_i}{\sum_{i=1}^{n} w_i x_i} < U_j \qquad (j=1...m) \qquad (1)$$

Multiplying by the denominator and carrying across the sign gives for the upper bound weight constraint

$$\sum_{i=1}^{n} w_i a_{ij} - U_j \sum_{i=1}^{n} w_i x_i < 0 \quad (j=1...m) \tag{2}$$

Collecting terms gives

$$\sum_{i=1}^{n} w_i (a_{ij} - U_j) x_i < 0 \quad (j=1...m) \tag{3}$$

Multiplying by -1 to reverse the sense of the inequality gives

$$\sum_{i=1}^{n} w_i (U_j - a_{ij}) x_i > 0 \quad (j=1...m) \tag{4}$$

Similarly, for the lower bound weight constraint

$$\sum_{i=1}^{n} w_i a_{ij} x_i - L_j \sum_{i=1}^{n} w_i x_i > 0 \tag{5}$$

or

$$\sum_{i=1}^{n} w_i (a_{ij} - L_j) x_i > 0 \tag{6}$$

The general form for total ingot weight is given by

$$W_l < \sum_{i=1}^{n} w_i x_i < W_u \tag{7}$$

For the upper bound this becomes

$$\sum_{i=1}^{n} w_i x_i - W_u < 0 \tag{8}$$

Multiplying by -1

$$W_u - \sum_{i=1}^{n} w_i x_i > 0 \tag{9}$$

For the lower bound

$$\sum_{i=1}^{n} w_i x_i > W_l \tag{10}$$

or

$$\sum_{i=1}^{n} w_i x_i - W_l > 0 \tag{11}$$

The general form of the objective function is usually given as cost minimization and is

$$\text{Min } Z = \sum_{i=1}^{n} c_i x_i \quad \text{where } c_i \text{ is cost}$$

This is not a relevant consideration for RF Enterprises since costs will be virtually the same no matter which pigs they blend. What they want to do is to use the lowest possible assay value of pig to produce ingots of Alloy A and B so that the higher assay pigs are available if they have a sudden ingot order to fill requiring a very high assay of iron. For this reason, the sum of iron assays will be used in the objective function, or

$$\text{Min } Z = \sum_{i=1}^{n} a_{ij} x_i \tag{12}$$

It is acknowledged that summing the iron assay values will produce an objective function value greater than 100 percent, which is clearly nonsense. However, if it is recognized that the purpose of the objective function is to drive the problem in light of the constraints, and that the effect of using iron assay in the objective function is to penalize the use of the higher assay pigs, this concern evaporates. In reality, the objective

function value is only a number to respond to by noting the input variables that were used to produce it.

Since RF Enterprise wishes to make ingots of Alloy A and B two problem formulations are required, but these differ only in the treatment of total ingot assay value. It should be remembered that for this particular formulation, n=1...60 and m=1...20 (since the lower bounds for tramp elements are all zero and it is impossible to get a lower value than this, these equations are extraneous).

The program that produces the objective function and the functional constraints for weight, assay, and upper bounds on tramp elements is called TFORM.SIT and is in Appendix A. This code also sorts the variables in the objective function from lowest to highest assay and orders the functional constraints from most tight to loosest. The initial read out of the constraints shows the last eleven to be superfluous since they contain all positive entries that are, of course, greater than zero. The final bedsheet for the problem formulation is in Plate 1.

IMPLICIT ENUMERATION

Implicit enumeration is the name for a class of branch-and-bound algorithms designed specifically for the case in which a variable, x_i, is required to be 0 or 1, and thus the vector corresponding to all of these variables is binary. It is therefore understood that a vector could assume all the values from (0,0,...0) to (1,1,...1) by successively adding "one" to the rightmost bit. This, however, would mean 2^n possible solution combinations, a number which quickly grows large. Fortunately, many of these combinations can be discarded by various tests.

There are many implicit enumeration techniques for solving 0,1 problems. The algorithm of Lawler and Bell (1966), however, appears to be the most appropriate for solving discrete input optimization problems with arbirtary functional constraints.

This method is applicable to any problem that can be put into the form:

$$\text{Min} \quad g_0(x)$$

such that $g_{11}(x)-g_{12}(x) > 0$

$g_{21}(x)-g_{22}(x) > 0$

$\vdots$

$g_{m1}(x)-g_{m2}(x) > 0$

where $x = (x_1, x_2, \ldots x_n)$

and $x_i = 0$ or 1 for $i = 1,2,\ldots,n$.

It is specified that each of the functions, g_0, g_{11},... g_{m2} is monotone nondecreasing in each of the variables x_1, $x_2,\ldots,x_n$ (Lawler and Bell, 1966). The functions g_{m1} and g_{m2} are determined by grouping the coefficients of like sign in each row of constraints. Any negative coefficients in the objective function can be made positive by allowing $x_i = 1-x'_i$ and carrying a constant.

Lawler and Bell makes use of a numerical ordering of solution vectors, where each vector is identified with an integer value. This ordering is precisely a lexicographically decreasing ordering of the binary n vectors (Garfinkle and Nemhauser, 1972).

The Lawler and Bell algorithm sets up the following rules that allow a considerable number of steps to be skipped:

Rule 1: If $g_0(x) > g_0(\hat{x})$, skip to x^*.

Rule 2: If $g_0(x) < g_0(\hat{x})$, and "x" is a feasible solution, skip to x*.

Rule 3: If $g_{m1}(x^*-1)-g_{m2}(x) < 0$, skip to x^*.

The value of x^* is given by the Boolean addition of (x.OR.(x-1))+1 or "x" plus the lowest ordered bit that is a "one" (Jansen, 1975).

Justification for Rules 1 and 2 is that the current best solution for the problem, x, is better than any solution between x and x^*, since any vector between x and x^* has at least one bit added to the right of the lowest ordered "one" and so only acts to increase the value of the objective function.

Justification for Rule 3 is a bit more involved. If a vector, x, is checked against a particular constraint which is not satisfied by x, x^*-1 is generated in order to see if turning on all the variables that have a positive coefficient to the lowest ordered "one" will satisfy the constraint. If this value still does not satisfy the constraint, the algorithm should continue from x^*, since this constraint will be unsatisfied in the vector range x through x^*-1.

An example from Lawler and Bell should serve to clarify this.

Given the functions

$$g_0(x_1,x_2,x_3,x_4)=3x_1+2x_2+x_3+x_4$$

$$-g_1(x_1,x_2,x_3,x_4)=-x_2x_3-x_4+1$$

$$g_2(x_1,x_2,x_3,x_4)=2x_1+x_2x_3+x_4-3$$

and $x_1,x_2,x_3,x_4 = 0$ or 1,

The skipping rules apply as follows:

x_1	x_2	x_3	x_4	
0	0	0	0	Skip to x^*=(0001), Rule 3
0	0	0	1	Skip to x^*=(0010), Rule 3
0	0	1	0	Skip to x^*=(0100), Rule 3
0	1	0	0	Skip to x^*=(1000), Rule 3
1	0	0	0	Infeasible, $g_2(x) < 0$
1	0	0	1	Feasible, skip to x^*=(1010), Rule 2
1	0	1	0	Skip to x^*=(1100), Rule 1
1	1	0	0	Skip to x^*=1(0000), Rule 1, Stop

By limiting the number of solutions to be examined, computation time can be significantly reduced. The other step to make this and any integer algorithm run more quickly is to order the constraints from tighest to most loose. The objective function constraints should be ordered from low to high for minimization problems.

This algorithm is coded in FORTRAN and written for a DECsystem 1091. This can be used to address the problem of RF Enterprises or any discrete blending problem if the problem warrents the considerable computer time it will take.

VECTOR ANALYSIS

A variation on the method of Senju and Toyoda can be used on 0 - 1 problems where the number of variables are great and the restricting conditions many. This appears to be the case for RF Enterprise. The fundamental concept of this approach is to use an ordinal scale to describe variables so that appropriate variables can be chosen so as to enter into a feasible space. This method is not directly guided by an objective function but rather influenced by the centroid of the feasible space. Before explaining this adaptation further, it will be necessary to review the method of Senju and Toyoda.

Senju and Toyoda is a conceptually simple method to understand because it can be pictured graphically. Consider the simple problem where two inputs, resource 1 and resource 2, are used to make different products sold for specified prices. Upper limits for the two resources can be drawn as in figure 1 to make a box.

Vectors for the product describe the use of the two resources. When these vectors are added together for all possible products made, the upper limits on resources are exceeded so it becomes necessary to drop products from the line until the vectors terminate within the box.

This is done by calculating the effective gradient (the ratio of the selling price to the sum of scalar products) for each vector and dropping off those products with the lowest effective gradients. When all vectors (products) are within the box, any surplus resources can be applied to add in formerly dropped products (Senju and Toyoda, 1968). In essence, the Senju and Toyoda method seeks to break out of the feasible region in order to come into it again based on some measure of goodness.

The adaptation to Senju and Toyoda is to establish a feasible region and, within this, a centroid point. For purposes here, the centroid is the average of acceptable weights and assays for a given ingot alloy but this centroid could be adjusted if it were found that there really is a normative value for these.

This algorithm choses an iron pig by looking at the least distance from the centroid to the vectors describing the weights and chemistries of the pigs. Once one pig is chosen, the axes are re-established at this point and the process repeats, again looking for the least distance between the vectors and the centroid. Figures 2 and 3 help to show this process in two dimensional space, but it should be recognized that this method will apply to n-dimensional space as well.

Figure 1: Senju and Toyoda

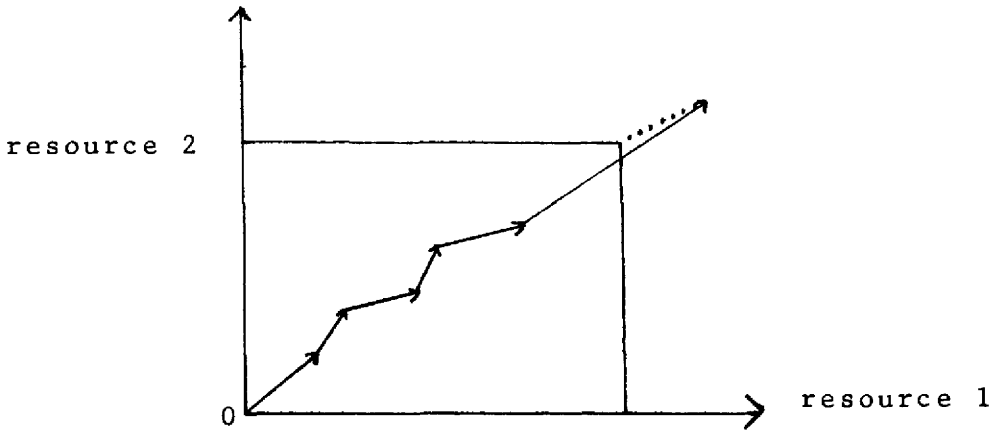


Figure 2: Adaptation

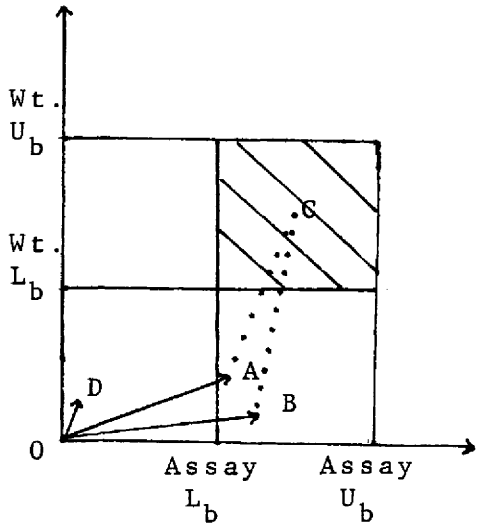
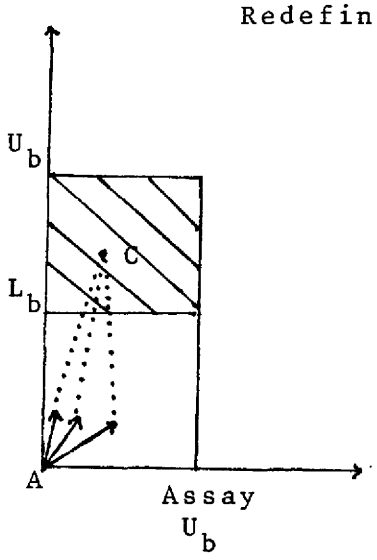


Figure 3: Axes Redefined



In figure 2, the feasible region is described by upper and lower bounds for assay and weight of an iron ingot and is shown as the diagonally slashed box. The centroid, C, is the average of the weights and assays. The distance from C to the vectors is calculated for each vector and the shortest distance is chosen, in this case AC. The axes are re-established at A, as is shown in figure 3, and the procedure is repeated until there are no more vectors that can be added to minimize the distance to C. If at any time the weight specifications are exceeded, the first vector is dropped, the axes redefined, and a new vector added using the same least-distance rule.

There are some things to be noted about this method. It produces all the mutually exclusive feasible solutions for iron pigs that could be blended to make ingots of Alloy A or B. The pulling effect of the centroid suggests that these combinations are close approximations of optimal but may not be strictly optimal. In addition, the need to minimize the distance between the centroid and the vectors dictates that the "fattest pigs" are chosen first. None of this is necessarily bad, though, since the goal is to use the available inventory. Those pigs that are excluded from selection indicate idle inventory that might be

included in future blends with new inventory shipments. If a high percentage of inventory consistently goes unused, it is a sign to RF Enterprise to request an alteration of specifications from their suppliers.

QUICK AND DIRTY ROUTINE

The heuristic of Riesco and Thomas (1968) was developed for linear programming problems that have homogeneous objective functions, finite upper bounds on problem variables and right hand side vector components that are all zero. This method is an iterative algorithm that starts with an initial known feasible solution vector. At each iteration the value of one of the components of the solution vector is increased, thus increasing the value of the objective function. The decision rule is to select the variable which will yield the greatest increase in the objective function value for that iteration. The basic premise of this heuristic is to see how much the variables in the constraints can be increased from the present position.

Modifications to this heuristic are made in light of the special concerns of RF Enterprise. First, while they would like to blend pigs to make ingots of lowest acceptable assay value, they would like even more to be able to use the full spread of their inventory subject to constraints on weight, assay and tramp elements. Therefore, the minimizing objective function will not be used. Secondly, historical experience of RF Enterprise states that at least three pigs sometimes four, possibly five,

must be used to produce one ingot. If they knew which pigs to choose to get a starting feasible solution, they could stop there but they don't know this. They do, however, know that the pigs closest in assay or approximately equidistant in assay from the ingot assay they wish to make will be good potential blends. Because of this knowledge, they could choose two pigs and solve for the requirements of the third. If the requirements for the third pig exceed those of any single available inventory, a forth pig must be chosen and so on. If the combination of all selected pigs exceeds the upper bound constraints for weight, assay or tramp elements, the pig that contributes most to this excess should be dropped and another one chosen. This need not be a complicated procedure and, in fact, can be done by a quick and dirty routine to be outlined .

Many integer problems appear to be formidable during formulation but have startling clarity if the user will simply look at the bedsheet for the problem. Plate 1 is the bedsheet for Alloy A and some things are apparent. All of these constraints are written as >0, so that any numbers added in any row must be >0. Assay values for pigs are listed from low to high. The tightest constraints

are 1-4. Constraints 6-9 must not be broken but can be given almost incidental consideration. Most likely blends will probably occur in the middle third of the data. Pigs chosen from the lower third of the data will probably need to be balanced by those from the upper third.

Supplied with this information, the user has enough data to do a systematic search for combinations of pigs to produce ingots. It is recognized that this search could become overwhelming if many ingots had to be produced each day, but if fewer than five or six ingots per day are needed, such a search routine could be eminently practical.

The quick and dirty routine is as follows:

1. Choose two pigs from Plate 1.
    a. Record the pig number to the left of rows a and b
    b. Record the constituent values for these pigs to the right of a and b. These values are taken from the bedsheet directly below the objective function.
2. Add vertically rows a and b for constituents 1-9. Put these values in row c.
3. Enter the right most value just before the inequality sign for each <u>constraint</u> in Plate 1 on line d.

4. Add rows c and d vertically for constituents 1-9. Put these values in row e.
5. Multiply each entry in row e by -1; Put these values in row f.
6. Look for a pig or pigs in Plate 1 that has value(s) greater than or equal to those listed for constituents 1-9 in row f. Put these values in row g and h.
7. Add rows g and h vertically for each constituent 1-9. Put in i.
8. Add rows i and e; Put in j. If all entries are positive numbers, STOP. The pigs listed in the left most column will produce an ingot in the required assay range.

The format for this routine is shown below.

Pig #	Row	Constraint 1 2 3 4 5 6 7 8 9

An application of this routine using Plate 1 data is shown in Table 4.

Pigs a and b are chosen in the middle third of the data. These appear to be good initial choices since they reduce the negativity of constraints 1 and 2 by about half and the negative value for the fifth constraint can be off-set with additional pigs. All the other constraints are positive.

The negative entries in row f of Table 4 cannot be exceeded by other negative additions. For example, those pigs that are added cannot have a value for constraint 1 of more than -2.8322.

Pigs g and h are chosen because they satisfy the first and second constraints. The negative elements for the third constraint is off-set by that of the 24th and 25th pigs.

The negative element in the eighth constraint is off-set by the sum of the corresponding constraints.

All entries in row j are positive therefore all constraints are satisfied.

Pig#	Row	Constraints 1	2	3	4	5	6	7	8	9
24	a	-1.0986	1.0986	.0028	.0002	-.9057	1.7010	1.5622	.7456	.0699
25	b	-.9892	.9892	.0015	.0012	-.2913	1.8200	1.4352	1.7511	1.4263
	c	-2.0878	2.0878	.0043	.0014	-1.1970	3.5210	2.9974	2.4967	1.4962
	d	4.9200	-3.8700	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	e	2.8322	-1.7822	.0043	.0014	-1.1970	3.5210	2.9974	2.4967	1.4962
	f	-2.8322	1.7822	-.0043	-.0014	1.1970	-3.5210	-2.9974	-2.4967	-1.4962
27	g	-.9405	.9405	-.0001	.0027	.9949	1.3483	.7000	1.3891	1.0782
29	h	-.9742	.9742	-.0009	.0036	.3761	1.8765	2.5651	-1.0268	.5889
	i	-1.9147	1.9147	.0008	.0063	1.3700	3.2248	3.2651	.3623	1.6671
	j	.9175	.1325	.0051	.0077	.1730	6.7458	6.2625	2.8590	3.1633

TABLE 4

Quick and Dirty

RESULTS

Lawler and Bell

It can be proven mathematically that the partial enumeration algorithm of Lawler and Bell will converge on the optimal solution for a blend. If five optimal solutions per day are required, the algorithm must be run five times with the previous variable combinations deleted from the data set each time. This is true for Alloy A or Alloy B. The impracticality of such an approach becomes apparent when it is realized that in six hours of computer run time using the data for Alloy A, this algorithm did not produce an optimal solution nor had all the variables even been enumerated. It seems reasonable to believe that the time required to find five optimal blends per day could exceed the time available in any work day to actually do the blending.

Vector Analysis

The modification to Senju and Toyoda produces mutually exclusive feasible blends for ingots of Alloy A or Alloy B in approximately twenty seconds of computer machine time. Three ingots can be made of Alloy A; five ingots can be made of Alloy B. The combination of iron pigs used to make these alloys are listed below by the pig variable number and correspond to those on Plates 1 and 2.

Alloy A		
Pig #'s:	20, 13, 9, 53	= 1 ingot
Pig #'s:	7, 2, 49, 16	= 1 ingot
Pig #'s:	4, 1, 47, 6	= 1 ingot

Alloy B		
Pig #'s:	53, 49, 47	= 1 ingot
Pig #'s:	40, 35, 51	= 1 ingot
Pig #'s:	52, 31, 54	= 1 ingot
Pig #'s:	60, 39, 41	= 1 ingot
Pig #'s:	32, 56, 50	= 1 ingot

Quick and Dirty

The quick and dirty approach has been shown by example to yield feasible solutions. The ease with which these are obtained is enhanced by a spread out bedsheet and familiarity with the data. In effect the user is attempting to add the row coefficients and right most constant for those pigs "turned on" to produce a non-negative number and thus satisfy all constraints.

CONCLUSIONS

It can be seen that the 0-1 problem of blending iron pigs to make ingots of a required assay value can be approached from the three directions outlined in this thesis. Only the rigorous method of implicit enumeration can guarantee optimal solutions if solutions exist, but there are no guarantees when that solution will occur. The vector analysis approach dealing with ordinal scales will produce feasible solutions quickly and with little computational burden. The quick and dirty scanning routine can be used to obtain feasible solutions if the bedsheet for the problem is laid out to reflect the relationships of the constituents to the constraints and many of the constraints are loose as is the case of the problen examined here.

Optimal solutions come at a price which can often be so high that the value of the solution is diminished. Computer costs, lost production time, and carrying costs for idle inventory must be weighed against the benefits of optimal solutions. On a cost basis, optimal solutions can prove to be a poor choice.

REFERENCES CITED

Hillier, Fredrick and Lieberman, Gerald, 1980, Introduction to Operations Research, Holden-Day Inc., London, San Francisco.

Garfinkel, Robert S. and Nemhauser, George L., 1972, Integer Programming, John Wiley and Sons, New York, London.

Jansen, Walrave T., 1975, Unpublished Thesis, Colorado School of Mines.

Riesco, Armando and Thomas, Michael E., 1969, "A Heuristic Solution Procedure for Linear Programming Problems with Special Structure," AIIE Transactions.

Salkin, Harvey M., 1975, Integer Programming, Addison-Wesley Publishing Company, Reading, Mass., Menlo Park, Calif.

Senju, Shizuo and Toyoda, Yoshiake, 1968, "An Approach to Linear Programming with 0-1 Variables," Management Science, Vol. 15, No. 4.

APPENDIX A

Transformation Code

```
      DIMENSION J0(22),X(18,60),XOUT1(22,62),XOUT2(22,62)
      DIMENSION B(16)
      READ (10,100)((X(J,I),I=1,60),J=1,18)
  100 FORMAT(G)
      READ (2,100)B
      READ(3,100)J0

      DO 90 I=1,60
          K=I
          DO 70 J=I,60
      IF(X(17,J).LE.X(17,K)) GO TO 70
      K=J
   70     CONTINUE
          DO 80 J=1,18
      SAVE=X(J,K)
      X(J,K)=X(J,I)
      X(J,I)=SAVE
   80     CONTINUE
   90 CONTINUE
      DO 1000 I=1,16
          I1=I+1
          DO 1000 J=1,60
      XOUT1(I1,J)=(B(I)-X(I,J))*X(18,J)
      XOUT2(I1,J)=XOUT1(I1,J)
 1000 CONTINUE
      DO 1100 I=1,60
          XOUT1(1,I)=X(17,I)
          XOUT2(1,I)=X(17,I)
 1100 CONTINUE
      DO 1200 J=1,60
          XOUT1(18,J)=(X(17,J)-0.9896)*X(18,J)
          XOUT1(19,J)=(1.0041-X(17,J))*X(18,J)
          XOUT2(18,J)=(X(17,J)-1.0041)*X(18,J)
          XOUT2(19,J)=(1.0356-X(17,J))*X(18,J)
          XOUT1(20,J)=X(18,J)
          XOUT1(21,J)=-X(18,J)
          XOUT2(20,J)=XOUT1(20,J)
          XOUT2(21,J)=XOUT1(21,J)
 1200 CONTINUE
          DO 1300 J=1,21
      XOUT1(J,61)=0
      XOUT2(J,61)=0
 1300 CONTINUE
      XOUT1(20,61)=3.87
      XOUT1(21,61)=-4.92
      XOUT2(20,61)=3.87
      XOUT2(21,61)=-4.92
      IOUT1=57
      IOUT2=58
      WRITE(IOUT1,9000)(XOUT1(1,I),I=1,60)
      WRITE(IOUT2,9000)(XOUT2(1,I),I=1,60)
```

```
      DO 1500 J=2,21
          WRITE(IOUT1,9100)(XOUT1(J,I),I=1,60),XOUT1(J,61)
          WRITE(IOUT2,9100)(XOUT2(J,I),I=1,60),XOUT2(J,61)
 1500 CONTINUE
 9000 FORMAT(61F8.4)
 9100 FORMAT(62F8.4)
      STOP
      END
```

APPENDIX B

Lawler and Bell Code

```
      implicit integer (a-z)
      real cnstrt(62,22),x(61),feas(61),optmum,sum
      dimension fspec(10)
      double precision day
      common n

      data brief,batch /0, 0/
      data ibatch,obatch,itty,otty,dsk,lpt /5, 6, 4, 4, 1, 6/
      data cmax1,cmax2,spclen /62, 22, 10/
      data objfun,cstart,cladj,c2adj /1, 2, 1, 1/
      data yes,no,lettx,ge,eq /1, 0, 1hX, 4H.GE., 4H.EQ./

5     format(' Enter the input data file name: ',$)
10    format(10a5)
15    format(65g)
20    format(' ? Too many terms in the objective function. The',
     *       ' max is',i3)
25    format(' ? Too many constraints. The max is',i3)
30    format(' Enter the output data file name: ',$)
35    format(' Place a nice title here. Date = ',a10)
40    format(18x,8x,a1,i2,8x,a1,i2,8x,a1,i2,8x,a1,i2,8x,a1,i2,
     *         8x,a1,i2,8x,a1,i2,8x,a1,i2,8x,a1,i2,8x,a1,i2)
45    format(' Objective Function:',10f11,/,(20x,10f11))
50    format(' Constraint No.',i3,2x,10f11,/,(20x,10f11))
55    format(27x,a4,f11,7x,a4,f11)
60    format(' Lower Limit:'14x,'0.',/,' Upper Limit:',5x,f11)
65    format(' Feasible Solution:')
70    format(' Optimum Solution:  ',f11)

      do 100 i=1,cmax1
      do 100 j=1,cmax2
      cnstrt(i,j)=0.
100   continue
      j=cmax1-cladj
      do 105 i=1,j
      x(i)=0
105   continue

      in=ibatch
      in2=ibatch
      out=obatch
      out2=obatch
      if(batch.eq.yes) goto 110

      in=itty
      in2=itty
      out=otty
      out2=otty

      write(out,5)
      read(in,10,end=235) fspec
```

```
      call clean(fspec,spclen)

      if(fspec(1).eq.'TTY:'.or.fspec(1).eq.' ') goto 110
      in2=dsk
      open(unit=in2,dialog=fspec,access='seqin')

110   read(in2,15,end=225) (cnstrt(i,objfun),i=1,cmax1)

      n=cmax1-cladj

      if(cnstrt(cmax1,objfun).eq.0) goto 115
      write(out,20) n
      callexit

115   if(cnstrt(n,objfun).ne.0) goto 120
      n=n-1
      if(n.gt.0) goto 115
      stop '? Too few terms in the objective function'

120   r=n+1
      m=cstart
125   if(m.gt.cmax2) goto 130
      read(in2,15,end=135) (cnstrt(i,m),i=1,n),cnstrt(r,m)
      m=m+1
      goto 125

130   read(in2,15,end=140)
      i=cmax2-c2adj
      write(out,25) i
      callexit

135   if(m.eq.cstart) stop '? Too few constraints'

140   m=m-1

      if(in2.eq.dsk) close(unit=dsk)
      write(4,*) m

      if(batch.eq.yes) goto 145

      if(brief.eq.yes) goto 155

      write(out,30)
      read(in,10,end=235) fspec
      call clean(fspec,spclen)

      if(fspec(1).eq.'TTY:'.or.fspec(1).eq.' ') goto 145

      out2=lpt
      if(fspec(1).eq.'LPT:') goto 145
```

```
      out2=dsk
      open(unit=out2,dialog=fspec,access='seqout')

145   call date(day)

      write(out2,35) day
      write(out2,40) (lettx,i,i=1,n)
      write(out2,15)
      write(out2,45) (cnstrt(i,objfun),i=1,n)

      do 150 i=cstart,m
      write(out2,15)
      k=i-c2adj
      write(out2,50) k,(cnstrt(j,i),j=1,n)
      write(out2,55) ge,cnstrt(r,i)
150   continue
      write(out2,15)
      write(out2,15)

155   optmum=0.
      do 160 i=1,n
      optmum=optmum+cnstrt(i,objfun)
160   continue

      if(out.ne.out2) write(out,60) optmum
      write(out2,60) optmum

165   do 180 i=cstart,m
      sum=0.
      do 175 j=1,n
      sum=sum+cnstrt(j,i)*x(j)
175   continue
      if(sum.lt.cnstrt(r,i)) goto 210
180   continue

      if(brief.eq.yes) goto 195

      write(out2,15)
      write(out2,15)
      write(out2,65)

      do 190 i=cstart,m
      sum=0.
      do 185 j=1,n
      feas(j)=cnstrt(j,i)*x(j)
      sum=sum+feas(j)
185   continue

      k=i-c2adj
      write(out2,50) k,(feas(j),j=1,n)
      write(out2,55) eq,sum,ge,cnstrt(r,i)
```

```
      write(out2,15)
190   continue

195   sum=0.
      do 200 i=1,n
      feas(i)=cnstrt(i,objfun)*x(i)
      sum=sum+feas(i)
200   continue

      if(sum.lt.optmum) optmum=sum

      if(brief.eq.yes) goto 205
      write(out2,45) (feas(i),i=1,n)
      write(out2,55) eq,sum

205   if(jmpone(x).eq.no) goto 165
      goto 215

210   if(addone(x).eq.no) goto 165

215   if(out.eq.out2) goto 220
      write(out2,15)
      write(out2,15)
      write(out2,70) optmum

220   write(out,15)
      write(out,15)
      write(out,70) optmum

      callexit

225   write(out,230) fspec
230   format(' ? Empty input file: ',10a5)

235   callexit
      end



      subroutine clean(buffer,words)

      implicit integer (a-z)
      dimension ch(100),buffer(1)
      data smalla,smallz,diff /"141, "172, "40/

5     format(100r1)

      chars=words*5
      if(chars.gt.100)
     *           stop '? Too many characters for Subroutine CLEAN'
      decode(chars,5,buffer) (ch(i),i=1,chars)
```

```
      do 10 i=1,chars
      if(ch(i).ge.smalla.and.ch(i).le.smallz) ch(i)=ch(i)-diff
 10   continue
      encode(chars,5,buffer) (ch(i),i=1,chars)
      return
      end


      integer function jmpone(x)
      implicit integer (a-z)
      real x(1)
      common xsize
      if(xsize.le.0) stop '? Must have positive number of digits'
      i=xsize
      do 5 j=1,xsize
      if(x(i).eq.1.) goto 10
      i=i-1
 5    continue
 10   k=i+1
      do 15 j=1,i
      x(j)=0.
 15   continue
      jmpone=1
      if(k.gt.xsize) return
      jmpone=0
      x(k)=1.
      return
      end


      integer function addone(x)
      implicit integer (a-z)
      dimension xx(100)
      real x(1)
      common xsize
      if(xsize.le.0) stop '? Must have a positive number of digits
      do 5 dig=1,xsize
      if(x(dig).ne.1.) goto 10
      x(dig)=0.
 5    continue
      addone=1
      return
 10   addone=0
      x(dig)=1.
C     do 100 i=1,xsize
C 100 xx(i)=x(i)
C     write(4,111) (xx(j),j=1,xsize)
C 111 format(1x,65i1)
      return
      end
```

APPENDIX C

Minimum Vector Distance Code

```
      DIMENSION LIST(7),A(19,60),R(60),CO(19),T(19),NAME(60)
      TYPE *,'WHAT IS THE LOGICAL UNIT NUMBER? '
      ACCEPT *,IN
      READ(IN,900)((A(J,I),I=1,60),J=1,19)
      NBUTNS=60
      U=4.92
      BL=3.87
      DO 70 I=1,60
          NAME(I)=I
   70 CONTINUE
   80 DO 90 I=1,7
          LIST(I)=0
   90 CONTINUE
      DO 95 I=1,19
          CO(I)=0.0
   95 CONTINUE
      ICOUNT=0
  100 HOLD=0
      TEST=-1000000.
      DO 110 I=1,NBUTNS
          R(I)=0
          DO 102 J=1,19
      R(I)=R(I)+A(J,I)**2+A(J,I)*CO(J)
  102     CONTINUE
          IF(R(I).LE.TEST) GO TO 110
          DO 105 J=1,6
      IF(I.EQ.LIST(J)) GO TO 110
  105     CONTINUE
          DO 107 J=1,19
      T(J)=CO(J)+A(J,I)
      IF(T(J).LT.0.)GO TO 110
  107     CONTINUE
          TEST=R(I)
          HOLD=I
  110 CONTINUE
      IF(HOLD.EQ.0) GO TO 115
      ICOUNT=ICOUNT+1
      IF(ICOUNT.GT.5) GO TO 115
      LIST(ICOUNT)=HOLD
      DO 111 I=1,19
          CO(I)=CO(I)+A(I,HOLD)
  111 CONTINUE
      IF(CO(19).GT.U)GO TO 115
      IF(CO(19).LT.BL) GO TO 100
      DO 1110 I=1,18
          IF(CO(I).LT.0)GO TO 115
 1110 CONTINUE
      TYPE 1000
      DO 112 I=1,6
          IF (LIST(I).EQ.0) GO TO 112
          TYPE 1100,NAME(LIST(I))
```

```
          N=I
  112 CONTINUE
      DO 30 I=1,N-1
          L=100
          DO 20 J=I,N
      IF(L.LE.LIST(J)) GO TO 20
      L0=J
      L=LIST(J)
   20     CONTINUE
          LIST(L0)=LIST(I)
          LIST(I)=L
   30 CONTINUE
      K=0
      L=1
      NBUTNS=NBUTNS-N
      IF(NBUTNS.LT.46) STOP
      DO 50 I=1,NBUTNS+N
          IF(I.NE.LIST(L)) GO TO 40
          L=L+1
          K=K+1
          GO TO 50
   40     IF(K.EQ.0) GO TO 50
          DO 45 J=1,19
      A(J,I-K)=A(J,I)
   45     CONTINUE
          NAME(I-K)=NAME(I)
   50 CONTINUE
      GO TO 80
  115 DO 117 I=1,19
          CO(I)=CO(I)-A(I,LIST(1))
  117 CONTINUE
      DO 120 I=1,6
          LIST(I)=LIST(I+1)
  120 CONTINUE
      ICOUNT=ICOUNT-1
      GO TO 100
  900 FORMAT(60G)
 1000 FORMAT(/,' FEASABLE SOLUTION:')
 1100 FORMAT(2I)
      END
```

Alloy – A

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	X_{36}	X_{37}	X_{38}	X_{39}	X_{40}	X_{41}	X_{42}	X_{43}	X_{44}	X_{45}	X_{46}	X_{47}	X_{48}	X_{49}	X_{50}	X_{51}	X_{52}	X_{53}	X_{54}	X_{55}	X_{56}	X_{57}	X_{58}	X_{59}	X_{60}		
Objective Function	0.9046	0.8852	0.9060	0.9061	0.9052	0.9062	0.9067	0.9072	0.9074	0.9076	0.9083	0.9085	0.9088	0.9091	0.9092	0.9097	1.0001	1.0005	1.0006	1.0009	1.0011	1.0012	1.0013	1.0014	1.0015	1.0026	1.0034	1.0042	1.0048	1.0053	1.0057	1.0076	1.0122	1.0126	1.0132	1.0132	1.0133	1.0136	1.0136	1.0137	1.0137	1.0137	1.0138	1.0139	1.0140	1.0140	1.0141	1.0142	1.0142	1.0147	1.0147	1.0148	1.0148	1.0149	1.0149	1.0150	1.0152	1.0157		
Constraint 1	−0.9626	−0.9656	−1.0679	−0.9543	−1.0977	−1.1343	−0.9762	−1.0363	−1.1701	−1.0198	−0.9696	−1.0642	−1.0082	−1.0764	−0.8521	−1.1327	−1.0260	−1.0019	−0.9802	−1.0686	−1.0324	−0.9177	−0.9357	−1.0286	−0.9852	−0.8865	−0.9405	−0.8386	−0.9742	−0.9807	−1.1015	−1.1500	−1.2558	−1.2763	−1.3359	−1.2398	−1.3033	−1.2815	−1.3134	−1.3223	−1.3231	−1.2656	−1.1600	−1.2701	−1.2681	−1.2895	−1.3131	−1.2340	−1.3281	−1.2845	−1.3546	−1.3547	−1.3767	−1.2307	−1.2778	−1.3125	−1.2631	−1.2693	−1.3006	−1.2713	−1.2877	−1.8700	)	0
Constraint 2	0.9626	0.9656	1.0679	0.9543	1.0977	1.1343	0.9762	1.0363	1.1701	1.0198	0.9696	1.0642	1.0082	1.0764	0.8521	1.1327	1.0260	1.0019	0.9802	1.0686	1.0324	0.9177	0.9357	1.0286	0.9852	0.8865	0.9405	0.8386	0.9742	0.9807	1.1015	1.1500	1.2558	1.2763	1.3359	1.2398	1.3033	1.2815	1.3134	1.3223	1.3231	1.2656	1.1600	1.2701	1.2681	1.2895	1.3131	1.2340	1.3281	1.2845	1.3546	1.3547	1.3767	1.2307	1.2778	1.3125	1.2631	1.2693	1.3006	1.2713	1.2877	−1.8700	)	0
Constraint 3	0.0092	0.0086	0.0096	0.0076	0.0087	0.0089	0.0072	0.0071	0.0073	0.0071	0.0065	0.0059	0.0053	0.0054	0.0042	0.0055	0.0048	0.0036	0.0034	0.0034	0.0031	0.0027	0.0028	0.0028	0.0015	0.0006	0.0006	−0.0000	−0.0006	−0.0009	−0.0016	−0.0035	−0.0105	−0.0107	−0.0115	−0.0122	−0.0114	−0.0120	−0.0121	−0.0125	−0.0126	−0.0127	−0.0121	−0.0124	−0.0127	−0.0130	−0.0123	−0.0134	−0.0130	−0.0143	−0.0138	−0.0137	−0.0142	−0.0137	−0.0162	−0.0165	−0.0149	0.0000	)	0
Constraint 4	0.0048	0.0044	0.0060	0.0062	0.0072	0.0075	0.0063	0.0073	0.0087	0.0087	0.0084	0.0085	0.0083	0.0092	0.0092	0.0092	0.0115	0.0058	0.0059	0.0058	0.0121	0.0119	0.0116	0.0116	0.0131	0.0129	0.0132	0.0137	0.0127	0.0131	0.0155	0.0287	0.0253	0.0283	0.0301	0.0316	0.0294	0.0305	0.0307	0.0315	0.0318	0.0320	0.0305	0.0354	0.0307	0.0358	0.0314	0.0320	0.0327	0.0316	0.0327	0.0322	0.0352	0.0321	0.0331	0.0330	0.0336	0.0336	0.0000	)	0
Constraint 5	0.5144	0.5928	−0.5545	0.5398	−0.4306	−0.5285	0.6253	−1.7595	−0.3854	−0.3997	−1.0668	−0.5333	0.7310	−0.3974	−1.4277	−2.3635	−1.0539	−1.2496	−1.4708	0.6261	−0.8070	−0.3644	−0.2245	−0.9057	−0.2393	−2.0917	0.5945	−2.7723	0.3761	−0.6749	1.3536	1.3538	1.3852	1.3274	1.3334	1.2885	1.3554	1.3330	1.3659	1.3751	1.3823	1.3773	1.1704	1.3289	1.3298	1.3411	1.3657	1.3458	1.3853	1.3359	1.4088	1.3569	1.4315	1.3423	1.3285	1.3645	1.3527	1.3527	1.3332	1.3352	0.2000	)	0
Constraint 6	1.4752	1.7355	0.0904	1.8088	1.6408	1.2096	1.8274	1.1772	0.9870	0.3364	2.0941	1.9335	0.0201	0.1521	−1.3123	0.4390	0.4910	1.1666	0.9793	1.1112	1.6888	1.1101	0.4969	1.7010	1.6208	0.9710	1.1483	0.0096	1.8765	0.4843	3.0790	3.0477	2.9283	3.0614	3.1737	2.9337	3.0715	3.0322	3.1169	3.1088	3.1374	2.9613	2.9610	2.9398	3.0220	3.0288	3.0421	3.0461	3.1294	3.0435	3.2088	3.0436	3.2773	3.0168	2.9323	3.0591	2.9873	3.1180	2.5444	2.9750	0.0000	)	0
Constraint 7	−0.2563	1.8438	0.7755	1.2644	2.0226	1.5928	1.3780	0.3408	2.7294	1.9261	−0.2940	1.8527	1.8724	0.8235	−0.6452	1.6806	2.4381	2.0575	1.2661	1.7122	1.8130	1.2561	1.3725	1.5622	1.4352	1.5628	0.7090	−0.8448	2.5401	0.6689	3.4490	3.4451	3.2257	3.3823	3.5506	3.2823	3.4538	3.3967	3.4892	3.5048	3.2222	3.3565	3.3290	3.3677	3.3505	3.4773	3.4798	3.4292	3.5196	3.4035	3.5868	3.4674	3.4682	3.4263	3.3951	3.4768	3.3525	3.4467	3.3683	3.4124	0.0000	)	0
Constraint 8	0.3961	0.7538	1.2271	2.5486	2.1854	1.9913	2.0237	1.6393	2.5387	0.7205	0.9439	1.8779	0.7437	0.7742	−0.2093	1.8879	2.2155	1.6054	1.2961	0.3953	0.7784	0.7915	1.4527	0.7405	1.7201	2.0163	1.1881	0.4779	−1.6068	2.2801	2.2751	2.3624	2.1958	2.2439	2.3936	2.2648	2.3728	2.4335	2.3737	2.1479	2.7997	2.2770	2.1870	2.1688	2.2376	2.4504	2.7168	2.3071	2.2735	2.6174	2.1686	2.7252	2.1978	2.4693	2.4958	2.5001	2.1478	0.0000	)	0
Constraint 9	1.5332	0.3758	0.7953	0.2934	1.2169	1.2953	0.4563	0.5259	1.3386	0.7936	1.3000	0.8583	1.1106	1.2907	0.7541	1.0744	1.2937	0.7743	1.2898	0.7969	1.2417	1.7708	1.3652	0.9939	1.4263	1.1429	1.0782	0.9463	0.5883	0.5814	1.8093	1.8948	−0.5833	1.2849	1.2044	1.4253	0.2864	1.5362	−0.5476	1.2268	1.0658	−0.1095	0.3531	1.0069	0.4832	0.7273	0.4145	0.7776	2.2640	1.9020	0.6278	0.8043	0.5855	17.471	0.9018	2.0185	0.2492	1.0196	1.6301	1.2122	0.0000	)	0

Alloy – B

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	X_{36}	X_{37}	X_{38}	X_{39}	X_{40}	X_{41}	X_{42}	X_{43}	X_{44}	X_{45}	X_{46}	X_{47}	X_{48}	X_{49}	X_{50}	X_{51}	X_{52}	X_{53}	X_{54}	X_{55}	X_{56}	X_{57}	X_{58}	X_{59}	X_{60}		
Objective Function	0.9946	0.9953	0.9960	0.9961	0.9962	0.9962	0.9967	0.9972	0.9974	0.9976	0.9983	0.9985	0.9988	0.9991	0.9992	0.9997	1.0001	1.0003	1.0006	1.0009	1.0011	1.0012	1.0013	1.0015	1.0026	1.0034	1.0043	1.0048	1.0051	1.0057	1.0076	1.0122	1.0126	1.0132	1.0132	1.0133	1.0133	1.0136	1.0136	1.0136	1.0137	1.0137	1.0137	1.0138	1.0139	1.0140	1.0140	1.0141	1.0142	1.0142	1.0147	1.0147	1.0148	1.0149	1.0148	1.0149	1.0149	1.0150	1.0155	1.0157		
Constraint 1	−0.0626	−0.0616	−1.0679	−0.9543	−1.0377	−1.1343	−0.9762	−1.0363	−1.0195	−1.0918	−0.9230	−1.0642	−1.0892	−1.0791	−0.8523	−1.1537	−1.0290	1.0019	0.9892	1.0686	1.0324	0.9977	0.9957	1.0986	0.9892	0.8868	0.9405	0.8196	0.9742	0.9307	1.3015	1.3000	1.2550	1.2763	1.3359	1.3350	1.3033	1.2818	1.3514	1.3323	1.3231	1.3228	1.2666	1.2630	1.2719	1.2691	1.2895	1.3131	1.2945	1.3281	1.2845	1.3546	1.3547	1.3767	1.2907	1.2778	1.3120	1.2651	−1.3906	−1.2713	−1.2877	6.9200	)	0
Constraint 2	0.9636	0.9636	1.0679	0.9543	1.0377	1.1343	0.9762	1.0363	1.0195	1.0918	0.9230	1.0642	1.0892	1.0791	0.8523	1.1537	1.0290	1.0019	0.9892	1.0686	1.0324	0.9977	0.9957	1.0986	0.9892	0.8868	0.9405	0.8196	0.9742	0.9307	1.3015	1.3000	1.2550	1.2763	1.3359	1.3350	1.3033	1.2818	1.3514	1.3323	1.3231	1.3228	1.2666	1.2630	1.2719	1.2691	1.2895	1.3131	1.2945	1.3281	1.2845	1.3546	1.3547	1.3767	1.2907	1.2778	1.3120	1.2651	1.3906	1.2713	1.2877	−3.9700	)	0
Constraint 3	0.0395	0.0390	0.0423	0.0377	0.0433	0.0447	0.0380	0.0398	0.0408	0.0415	0.0357	0.0395	0.0371	0.0393	0.0310	0.0405	0.0364	0.0352	0.0343	0.0371	0.0355	0.0376	0.0342	0.0374	0.0326	0.0286	0.0285	0.0259	0.0257	0.0233	0.0312	0.0305	0.0288	0.0286	0.0300	0.0275	0.0289	0.0282	0.0285	0.0231	0.0292	0.0278	0.0275	0.0277	0.0276	0.0279	0.0284	0.0279	0.0284	0.0274	0.0283	0.0273	0.0287	0.0268	0.0266	0.0271	0.0281	0.0267	0.0251	0.0259	0.0000	)	0
Constraint 4	−0.0092	−0.0096	−0.0096	−0.0076	−0.0097	−0.0089	−0.0072	−0.0071	−0.0071	−0.0071	−0.0055	−0.0059	−0.0053	−0.0054	−0.0042	−0.0055	−0.0041	−0.0036	−0.0034	−0.0034	−0.0031	−0.0027	−0.0028	−0.0028	−0.0015	−0.0006	0.0001	0.0006	0.0003	0.0016	0.0098	0.0155	0.0107	0.0196	0.0122	0.0114	0.0120	0.0131	0.0125	0.0126	0.0127	0.0121	0.0121	0.0123	0.0124	0.0127	0.0130	0.0125	0.0134	0.0139	0.0143	0.0138	0.0147	0.0142	0.0137	0.0142	0.0145	0.0149	0.0000	)	0
Constraint 5	0.5344	0.5826	−3.5545	0.0398	−6.6306	−0.5281	0.0203	−1.7985	−0.3894	−0.3597	−1.0668	−0.5333	0.7310	−0.3974	−1.4277	−2.3636	−0.8539	−1.1416	−1.4708	0.0267	−0.8070	−3.3644	−0.2245	−0.9057	−0.2393	−2.0117	0.9940	−2.7723	0.3761	−0.6749	1.3536	1.3528	1.3052	1.3274	1.3934	1.3985	1.3554	1.3339	1.3659	1.3791	1.3823	1.3973	1.3954	1.3209	1.3158	1.3411	1.3657	1.3456	1.3813	1.3359	1.4088	1.3569	1.4316	1.3423	1.3289	1.3645	1.3527	1.3527	1.3228	1.3392	0.0000	)	0
Constraint 6	1.4792	1.7305	0.9804	1.5088	1.6460	1.2086	1.8274	1.3773	0.9870	0.9364	2.0941	1.5935	0.9395	0.2521	−1.3523	2.4388	0.4910	1.1966	0.3769	1.1912	1.6868	1.1101	0.4989	1.7905	1.6200	0.9765	1.5483	0.8986	1.8765	0.4841	1.9776	2.0477	2.3283	1.9034	2.1727	2.3337	2.0725	2.0522	2.1160	2.1068	2.1074	2.0613	2.0630	2.0991	2.0220	2.0288	2.0521	2.0413	2.1294	2.0435	2.0588	2.0436	2.2773	2.0968	2.2929	2.0501	2.0803	2.0185	2.2444	2.0750	0.0000	)	0
Constraint 7	−0.7663	1.8438	0.7755	1.9644	2.0216	1.5928	1.3769	0.1408	2.7234	1.3261	−0.7146	1.8127	1.8724	0.8215	−0.4452	1.6996	2.4381	2.0975	1.9661	1.7722	1.8130	1.1561	1.3725	1.5622	1.4352	1.5628	0.7000	−0.8441	2.5451	0.6683	3.4490	3.4451	3.3257	3.3823	3.5006	3.2833	3.4538	3.3967	3.4895	3.5940	3.5266	3.3393	3.3617	3.3605	3.4773	3.4798	3.4282	3.5196	3.4639	3.5808	3.4274	3.5482	3.4283	3.3961	3.4768	3.5525	3.4467	3.3689	3.4124	0.0000	)	0
Constraint 8	0.5961	0.7238	1.3271	2.3486	2.7814	1.9813	2.0297	1.6295	2.5187	0.7201	0.9438	1.8770	0.7437	0.7722	−0.2193	1.8879	2.1935	1.6302	1.1965	0.9363	0.7784	0.7113	1.4127	0.7406	1.7511	2.1463	1.3891	0.5479	−1.0268	2.2801	2.2775	2.0624	3.1058	3.1225	2.9595	2.3648	2.8729	2.4335	3.2737	2.1270	2.7017	2.3770	2.8870	2.3688	2.8976	2.4504	2.7168	3.3071	2.7735	2.6174	2.9605	2.7012	3.1078	2.8794	3.2169	2.7711	2.4953	2.4958	3.0901	3.1478	0.0000	)	0
Constraint 9	1.5932	0.3758	0.7653	0.2094	1.2189	1.0163	0.4565	0.8423	1.3186	0.7035	1.3900	0.8583	1.0106	1.8907	0.7641	1.0744	1.5957	0.7143	1.2698	0.7969	1.5417	1.7708	1.3602	0.8699	1.4263	1.1423	1.0782	0.8463	0.5883	0.3914	1.8093	1.8848	−0.5833	1.2649	1.2044	1.4453	0.2464	1.5362	−0.8476	1.2108	1.0618	−0.9565	0.3531	1.0069	0.4032	0.7275	0.4941	0.3776	2.2640	1.9920	0.6276	0.8043	0.8855	1.7471	0.5978	1.0985	0.2492	1.0196	1.6301	1.2122	0.0000	)	0