Article

Synthesizing Optimal Waste Blends

Venkatesh Narayan, Urmila M. Diwekar, and Mark Hoza

Ind. Eng. Chem. Res., **1996**, 35 (10), 3519-3527• DOI: 10.1021/ie960028c • Publication Date (Web): 08 October 1996

Synthesizing Optimal Waste Blends

Venkatesh Narayan and Urmila M. Diwekar*

Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

Mark Hoza

Pacific Northwest Laboratory, Battelle Boulevard, Richland, Washington 99352

Vitrification of tank wastes to form glass is a technique that will be used for the disposal of high-level waste at Hanford. Process and storage economics show that minimizing the total number of glass logs produced is the key to keeping cost as low as possible. The amount of glass produced can be reduced by blending of the wastes. The optimal way to combine the tanks to minimize the volume of glass can be determined from a discrete blend calculation. However, this problem results in a combinatorial explosion as the number of tanks increases. Moreover, the property constraints make this problem highly nonconvex where many algorithms get trapped in local minima. In this paper we examine the use of different combinatorial optimization approaches to solve this problem. A two-stage approach using a combination of simulated annealing and nonlinear programming (NLP) is developed. The results of different methods such as the heuristics approach based on human knowledge and judgment, the mixed integer nonlinear programming (MINLP) approach with GAMS, and branch and bound with lower bound derived from the structure of the given blending problem are compared with this coupled simulated annealing and NLP approach.

1. Introduction

The Hanford site in southeastern Washington produced nuclear materials using various processes for nearly 50 years. Radioactive waste was produced as byproducts of the processes. The current plan is for the waste to be retrieved and separated into high-level and low-level portions. The high-level and low-level wastes will be immobilized for future disposal. Retrieval, separation, and immobilization will all be complex, challenging, and time-consuming efforts.

The current plan is for high-level waste to be converted into a glass form for disposal. The glass must meet both processability and durability restrictions. The processability conditions ensure that the glass has properties such as viscosity, electrical conductivity, and liquidus temperature within ranges known to be acceptable for the vitrification process. Durability restrictions ensure that the resultant glass meets the quantitative criteria for disposal/long-term storage in a repository. There are also bounds on the composition of various components in the glass.

In the simplest case, waste and appropriate glass formers (frit) are mixed and heated in a melter to produce a glass that satisfies the processability and durability constraints. For a given waste, a wide range of glasses which satisfy those constraints can be formulated. If glasses are formulated to minimize the volume of glass that would be produced, then the cost of processing the waste and storing the resultant glass would be greatly reduced. Minimizing the volume of glass and minimizing the frit to be added (since glass will consist of waste and frit) are computationally equivalent.

Hanford has 177 tanks (50 000 to 1 million gallons) containing radioactive waste. Since these wastes result from a variety of processes, these wastes vary widely in composition, and the glasses produced from these wastes will be limited by a variety of components. The minimum amount of frit would be used if all the high-level wastes were combined to form a single feed to the vitrification process. Because of the volume of waste involved and the time span over which it will be processed, this is physically impossible. However, much of the same benefits can be obtained by forming blends from sets of tanks. The problem is how to divide all the tanks into sets to be blended together such that the minimal amount of frit is required.

In refinery and other processes blending is commonly used. There are several papers in the literature which report application of linear, successive linear and nonlinear programming, and mixed-integer programming methods to blending (Palacios-Gomez, 1980; Lasdon and Warren, 1980; Haverly, 1978; Thomas et al., 1978). Brief discussions related to these different approaches to blending problems are provided in Reklaitis et al. (1983) and Edgar and Himmelblau (1988). However, the highly nonconvex nature and computationally intensive nature of the discrete blending problem at hand made it necessary to apply various existing combinatorial optimization algorithms and a heuristic method. Therefore, a unique contribution of this paper is the comparison of various combinatorial optimization algorithms to a complex real world problem of significance.

In this discrete blending problem, there are N different sources of waste which have to form a discrete number of blends B, the number of blends being less than the number of sources or tanks. All the wastes from any given tank are required to go to a single blend, and each blend contains waste from N/B sources. Blends of equal size (same number of wastes per blend) were specified; alternatively, blends could be formulated to have approximately the same waste masses. If neither of these were specified as contraints, all the wastes would go to a single blend. Figure 1 shows a set of four wastes which need to be partitioned into two parts to form two blends. To find the optimal waste blends, all possible combinations of tanks into blends

* Author to whom correspondence is addressed at the Department of Engineering & Public Policy, Carnegie Mellon University. Phone: (412)268-3003. Fax: (412)268-3757. Email: ud01@andrew.cmu.edu.

S0888-5885(96)00028-0 CCC: $12.00

have to be examined. The number of possibilities grows exponentially in the size of the problem. Consequently, it becomes computationally impossible to solve this problem when the number of sources is large, as usually occurs in practice.

The state of the art combinatorial optimization techniques which can be used for the solution of the optimal waste blending problems include (a) the mathematical programming approach which involves the use of mixed-integer nonlinear programming (MINLP) algorithms, (b) the heuristic approach which relies on intuition and engineering knowledge, and (c) an approach based on the use of probabilistic optimization methods like simulated annealing.

In this paper we explore the use of the above-mentioned three approaches to solve this problem for a subset of the 177 tanks. For this study 21 tanks were chosen, which need to form three blends. At first, the problem was formulated and implemented in General Algebraic Modeling Systems (GAMS) (Brooke et al., 1992). GAMS uses the outer approximation/equality relaxation/augmented penalty function (OA/ER/AP) algorithm for the solution of MINLP problems. While this algorithm is computationally efficient, due to the nonconvex nature of constraints it fails to find the global optimum. A branch and bound procedure using the structures of the problem was also developed. Branch and bound is a well-known technique used to solve combinatorial problems (Morton and Pentico, 1993; Rich and Knight, 1991; Taha, 1975). This algorithm reached the global optimum but was computationally very intensive. The heuristic approach could get to a solution reasonably close to global optimum but was again restricted to this particular problem only. A two-stage approach involving combination of simulated annealing and nonlinear programming algorithms led to the global optimal solution in a reasonable amount of time.

The paper is organized as follows: Section 2 provides the problem definition and describes the nonlinear programming problem which needs to be solved for the mathematical programming approaches and for the two-stage combined simulated annealing and NLP approach. Section 3 is devoted to the mixed-integer nonlinear programming (MINLP) approaches based on the OA/ER/AP algorithm. Section 4 introduces the two-stage approach. Section 5 describes the simulated annealing method and its application for the problem on hand. A branch and bound procedure developed for this problem is presented in section 6. This procedure provides the global optimality check for the two-stage approach. Section 7 presents the conclusions.

2. Problem Description

The basic glass formulation problem was described in Hoza (1994). Here, we address the formulation of the discrete blending problem used in this study. Let N be the number of wastes, B be the specified number of blends, and T be the number of wastes per blend. Therefore,

$$T = N/B \tag{1}$$

B is always defined in a manner such that T is an integer. Frit is added to each of these blends to form glass of particular quality. If $w^{(i)}$ is the mass of the ith component in the waste which is obtained by adding the component mass in each tank given by $w_t^{(i)}$, $f^{(i)}$ is the mass of the ith component in the frit, and $g^{(i)}$ is the mass of the ith component in the glass (where the total mass of glass is given by G), there results the following equality constraints.

$$g^{(i)} = w^{(i)} + f^{(i)} \tag{2}$$

$$G = \sum_{i=1}^{n} g^{(i)} \tag{3}$$

$$p^{(i)} = g^{(i)}/G \tag{4}$$

where n is the total number of chemical components and $p^{(i)}$ denotes the fraction of the ith component in the glass. Note that G is composed of a known component, $w^{(i)}$, and an unknown component, $f^{(i)}$.

In order to form glass the blend must satisfy certain constraints. These constraints are briefly described below. A more detailed description of the constraints is provided in the appendix.

1. Individual Component Bounds: There are upper ($p_{UL}^{(i)}$) and lower ($p_{LL}^{(i)}$) limits on the fraction of each component in glass. Therefore,

$$p_{LL}^{(i)} \le p^{(i)} \le p_{UL}^{(i)} \tag{5}$$

2. Crystallinity Constraints: The crystallinity constraints or multiple-component constraints specify the limits on the combined fractions of different components. There are five such constraints.

3. Solubility Constraints: These constraints limit the maximum value for the mass fraction of one or a combination of components.

4. Glass Property Constraints: These constraints govern the properties of viscosity, electrical conductivity, and durability.

Blending is most effective when the limiting constraint is one of the first three types. In these cases, one, or a small number of components is limiting. The compositions of these components vary widely in the wastes, so blending averages out the limiting species, resulting in a better overall solution requiring less frit.

For a single blend the problem is a relatively straightforward nonlinear programming problem (NLP) with a linear objective and a mix of linear and quadratic constraints. The objective is to:

$$\text{Min } G \equiv \text{Min} \sum_{i=1}^{n} f^{(i)} \tag{6}$$

Subject to:

Equality constraints (eqs 2−4)
Individual component bounds (eq 5)
Crystallinity constraints
Solubility constraints
Glass property constraints
Nonnegativity constraints

The nonnegativity constraints are present because frit can only be added to comply with the constraints. Filtering out individual components of the waste to satisfy constraints is not an option. The problem is complicated by the presence of a set of wastes which need to be partitioned to form blends. Frit is added to each of the blends to form glass. Since the composition is different in each blend, the manner in which the waste set is partitioned affects the amount of frit which

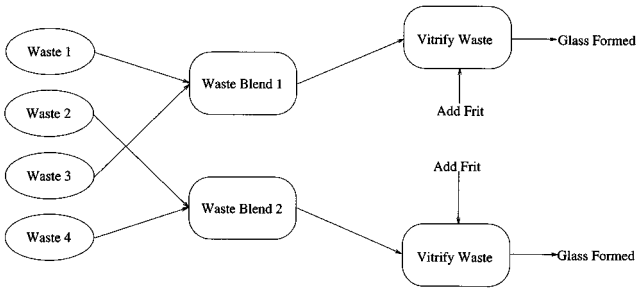

Figure 1. Conversion of waste to glass.

needs to be added. The objective continues to be to minimize the total amount of waste that needs to be vitrified. This is equivalent to minimizing the *sum* of frit added to each of the *B* blends. The problem stated above can be reformulated to reflect this:

$$\text{Min } G \equiv \text{Min} \sum_{j=1}^{B} \sum_{i=1}^{n} f_j^{(i)} \tag{7}$$

Subject to:

Equality constraints (eqs 2−4)
Individual component bounds (eq 5)
Crystallinity constraints
Solubility constraints
Glass property constraints
Nonnegativity constraints

for each blend j where $j = 1, ..., B$.

Figure 1 shows four wastes which are to be combined to form two blends of two wastes each. There are three possible combinations which must be examined in order to determine which one minimizes the total amount of frit required.

The objective in this phase is to select the combination of blends so that the total amount of frit used is minimized. The number of possible combinations is given by the formula:

$$\frac{N!}{B!(T!)^B} \tag{8}$$

The formula indicates the complexity of the problem. To put this into perspective, if, for example, there are six tanks which have to be combined to form two blends by combining three tanks, there are 10 possible combinations. If the number of individual waste tanks is 24 and four blends are to be formed by combining any six tanks, the number of possible combinations is 96 197 645 544. If the number of wastes is further quadrupled to 96 while maintaining the ratio of blends to the number of wastes in a blend at $^2/_3$, the number of possible combinations is approximately 8.875×10^{75}. Clearly, any approach which is required to examine every possible combination to guarantee the optimal will very quickly be overwhelmed by the number of possible choices. Further, note that a change in the ratio of the blends available to the number of wastes combining to form a blend affects the number of possible combinations. Figure 2 shows this variation when the number of wastes is 128. On the *x*-axis the number of blends formed increases from left to right and the number of wastes in a blend decreases from left to right. The *y*-axis represents the *log* of possible combinations. Notice that the number of combinations first increases and then decreases and is skewed somewhat to the right.

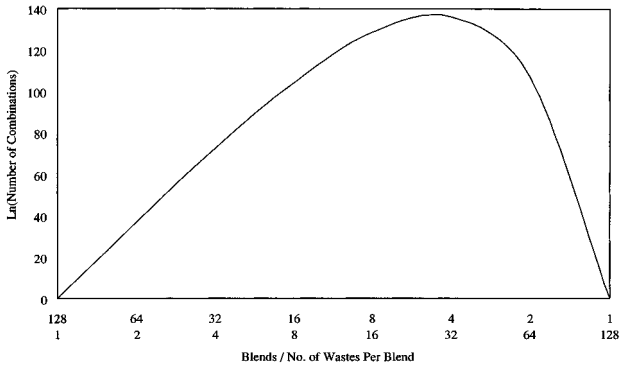

Figure 2. Combinatorial complexity versus number of blends.

For the purposes of this study we have selected 21 tanks to be partitioned into three blends. The information about chemical composition and amount of waste in each tank was obtained from the unpublished material provided by one of the authors and is presented in Table 1. From the above formula, for a problem with 21 wastes to be partitioned into three blends, there are 66 512 160 possible combinations to examine. Clearly examining all possible combinations is a very onerous task and nearly impossible for larger problems. We therefore either have to resort to a heuristic approach or use combinatorial optimization methods like mathematical programming techniques or simulated annealing.

In a heuristic approach to solving the discrete blending problem, we first determined the limiting constraint for a total blend of all tanks being considered (21, in this case). Then we tried to formulate blends such that all blends have the same limiting constraint. If this can be achieved, the frit required would be the same as that for the total blend. This approach, however, was very difficult to implement; rather, we formulated blends to try to ensure that all blends were near the limiting value for the limiting constraint. Using this approach, the best solution obtained was 11 736 kg of frit with the following tank configurations in each blend.

Blend 1

tanks = [15 11 14 17 5 12 8]

Blend 2

tanks = [18 20 6 3 7 4 19]

Blend 3

tanks = [21 1 10 16 3 2 9]

3. MINLP Approach

One possible approach for solving the above is using mixed-integer nonlinear programming (MINLP). Algorithms for solving MINLP include the generalized Benders decomposition (GBD) method (Benders, 1962; Geoffrion, 1972), the outer approximation (OA) method (Duran and Grossmann, 1986), and the branch and bound method (Beale, 1977; Gupta, 1980). The GBD and OA algorithms are, in general, more efficient than the branch and bound method (the branch and bound algorithm requires good lower bounds for faster convergence). These algorithms for solving the above MINLP consist of solving at each major iteration, an NLP subproblem (with all 0−1 variables fixed) and a mixed-integer linear programming (MILP) master prob-

Table 1. Waste Composition

	fractional composition of wastes $w^{(i)}/g^{(i)}$										
component	AY-102	AZ-101	AZ-102	SY-102	SY-101	SY-103	B-103	BY-104	BY-110	C-103	C-105
SiO_2	0.072	0.092	0.022	0.020	0.000	0.019	0.011	0.030	0.040	0.412	0.359
B_2O_3	0.026	0.000	0.006	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Na_2O	0.105	0.264	0.120	0.154	0.300	0.230	0.100	0.082	0.089	0.006	0.012
Li_2O	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CaO	0.061	0.012	0.010	0.030	0.007	0.006	0.000	0.141	0.046	0.041	0.044
MgO	0.040	0.000	0.003	0.012	0.000	0.001	0.000	0.000	0.000	0.028	0.026
Fe_2O_3	0.328	0.323	0.392	0.133	0.000	0.039	0.155	0.067	0.051	0.338	0.064
Al_2O_3	0.148	0.157	0.212	0.318	0.659	0.546	0.214	0.344	0.462	0.057	0.372
ZrO_2	0.002	0.057	0.063	0.002	0.000	0.001	0.000	0.007	0.003	0.043	0.004
other	0.217	0.096	0.173	0.328	0.034	0.159	0.520	0.330	0.309	0.075	0.119
total	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Cr_2O_3	0.016	0.007	0.005	0.089	0.002	0.116	0.000	0.000	0.000	0.002	0.005
F	0.006	0.001	0.001	0.005	0.002	0.001	0.000	0.001	0.001	0.000	0.000
P_2O_5	0.042	0.001	0.021	0.088	0.013	0.005	0.037	0.016	0.022	0.013	0.012
SO_3	0.001	0.018	0.009	0.027	0.005	0.002	0.007	0.002	0.003	0.000	0.002
NobMet	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
waste mass	59772	40409	143747	359609	167510	185990	6170	155473	103492	85211	207127

	fractional composition of wastes $w^{(i)}/g^{(i)}$									
component	C-106	C-108	C-109	C-111	C-112	S-102	SX-106	TX-105	TX-118	U-107
SiO_2	0.437	0.001	0.001	0.002	0.001	0.000	0.033	0.010	0.060	0.008
B_2O_3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Na_2O	0.014	0.010	0.007	0.011	0.005	0.337	0.280	0.168	0.425	0.038
Li_2O	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CaO	0.046	0.000	0.737	0.426	0.593	0.000	0.000	0.000	0.000	0.000
MgO	0.031	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Fe_2O_3	0.214	0.206	0.003	0.042	0.002	0.023	0.102	0.167	0.026	0.077
Al_2O_3	0.168	0.693	0.013	0.256	0.097	0.582	0.388	0.595	0.240	0.650
ZrO_2	0.008	0.032	0.000	0.007	0.000	0.000	0.000	0.000	0.000	0.000
other	0.082	0.058	0.238	0.256	0.302	0.058	0.197	0.060	0.250	0.228
total	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Cr_2O_3	0.004	0.002	0.000	0.000	0.000	0.024	0.020	0.000	0.000	0.000
F	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.004	0.001
P_2O_5	0.031	0.047	0.003	0.012	0.005	0.006	0.038	0.002	0.159	0.020
SO_3	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.001	0.009	0.001
NobMet	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
waste mass	367165	46919	53271	24485	65673	36537	45273	42200	412495	11504

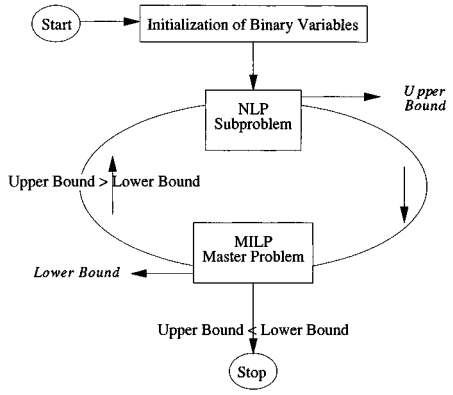

Figure 3. Main steps in GBD and OA algorithms for MINLP.

lem as shown in Figure 3. The NLP subproblem has the role of optimizing the continuous variables and providing the upper bound to the MINLP solution. The MILP master problem predicts the lower bound to the MINLP as well as updates the values of the 0−1 variables for each major iteration. The predicted lower bounds increase monotonically as the cycle of major iterations proceeds, and the search is terminated when the predicted lower bound coincides or exceeds the current upper bound.

The main difference between GBD and OA is in the definition of the master problem. In general, the OA algorithm requires fewer iterations than GBD but involves the solution of a larger master problem. Recent variants of these algorithms include the outer approximation/equality relaxation (OA/ER) strategy of Kocis and Grossmann (1987), extensions of GBD using a partitioning strategy (Floudas et al., 1989), the augmented penalty function/OA/ER algorithm (OA/ER/AP) of Viswanathan and Grossmann (1990), and the GBD/OA/ER/AP algorithm of Diwekar et al. (1992). GAMS uses the OA/ER/AP algorithm for solution of the resulting MINLP problem.

The GAMS-based MINLP solution was very dependent on the starting conditions for the calculation. The conditions specified were the initial distribution of each tank among the blends (for the relaxed initial optimization) and the frit composition of each of the blends. The best MINLP solution was found to be 12 342 kg of frit with the following blend composition.

Blend 1

tanks = [4 8 9 12 13 19 21]

Blend 2

tanks = [1 2 7 14 15 17 18]

Blend 3

tanks = [3 5 6 10 11 16 20]

The GAMS-based MINLP model failed to find the global optimal solution because the problem is highly nonconvex with the presence of several bilinear constraints.

For the particular problem in hand we also developed branch and bound procedure. Since this procedure was

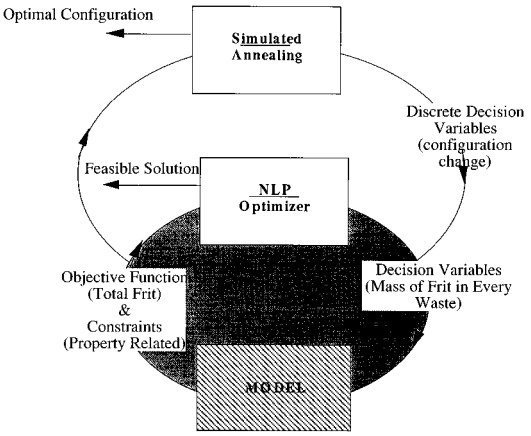

Figure 4. Optimum discrete blending problem: solution procedure.

specific to the three blend problem and also computationally intensive, it was used to check the global optimality of the simulated annealing solution procedure. Hence, it is presented as a separate section.

4. Two-Stage Approach

The optimal waste-loading problem which we have addressed here is the discrete blending problem, where the amount of frit required to meet the various constraints is minimized by blending optimal configurations of tanks and blends. We have used a two-loop solution procedure based on simulated annealing and nonlinear programming. In the inner loop nonlinear programming (NLP) is used to ensure constraint satisfaction by adding frit. In the outer loop the best combination of blends is sought using simulated annealing so that the total amount of frit used is minimized. Figure 4 shows the schematic of this procedure used for solving the discrete blend problem.

4.1. Inner Loop—NLP. Recent advances in constrained nonlinear optimization techniques provide better choices for solving the above NLP (Gill et al., 1981, 1987). The most popular of these methods are generalized reduced gradient (GRG) and successive quadratic programming (SQP) and their variants.

Among generalized reduced gradient methods, the most widely used algorithms are GRG2 and MINOS. Most literature on large-scale optimization for nonlinear data reconciliation favors the SQP method because the GRG2 algorithm requires convergence of equality constraints at each iteration. Historically, the infeasible path strategy was proposed to be the most efficient mode of optimization (Biegler, 1983). Hence, the GRG2 algorithm is not considered for this problem. On the other hand, MINOS does not require convergence of equality constraints but is best suited for optimization problems with linear constraints. Since there are nonlinear constraints in our formulation (please see the appendix), we are using SQP (Biegler and Cuthrell, 1985).

SQP is the most widely used technique for large-scale nonlinear optimization for chemical processes, which typically involve highly nonlinear models. In SQP, at each iteration the problem is approximated as a quadratic program where the objective function is quadratic and the constraints are linear. Similar to linear programming, the special features of a quadratic objective function are exploited to solve the problem more efficiently. The quadratic programming subproblem is solved for each step to obtain the next trial point. This cycle is repeated until the optimum is reached.

We have used the inner loop NLP to solve each single blend problem for both the two-stage approach and the branch and bound approach.

4.2. Outer Loop. The inner loop returns back the minimum amount of frit required to satisfy all the constraints given by eqs 2−5. We have used two different procedures for the outer loop. Section 5 describes the simulated annealing procedure, and section 6 describes branch and bound algorithm. Due to the problem characteristics, the solution from the inner loop cannot be guaranteed to be globally optimal. However, we are using the same NLP inner loop for the two-stage and the branch and bound approaches to find the discrete decision variables, i.e., the configuration of each blend. The branch and bound provides a guaranteed global optimum for the search of the discrete variables.

5. Simulated Annealing

Simulated annealing is a recently developed probabilistic method for combinatorial optimization based on ideas from statistical mechanics. The analogy in simulated annealing is to the behavior of physical systems in the presence of a heat bath: in physical annealing, all atomic particles arrange themselves in a lattice formation that minimizes the amount of energy in the substance, provided the initial temperature (T_{init}) is sufficiently high and the cooling is carried out slowly. At each temperature T, the system is allowed to reach thermal equilibrium, which is characterized by the probability (Pr) of being in a state with energy E given by the Boltzmann distribution:

$$Pr_E = \frac{1}{Z_t} e^{-\Delta E/K_b T} \tag{9}$$

where K_b is Boltzmann's constant (1.3806×10^{23} J/K) and $1/Z_t$ is a normalization factor (Collins et al., 1988).

In simulated annealing, the objective function (usually cost) becomes the energy of the system. The goal is to minimize the cost/energy. Simulating the behavior of the system then becomes a question of generating a random perturbation that displaces a "particle" (moving the system to another configuration). If the configuration S representing the set of the decision variables θ that results from the move has a lower energy state, the move is accepted. However, if the move is to a higher energy state, the move is accepted according to the Metropolis criteria (accepted with probability $= (1/Z_t)e^{-\Delta E/K_b T}$) (van Laarhoven and Aarts, 1987). This implies that, at high temperatures, a large percentage of uphill moves is accepted. However, as the temperature gets colder, a small percentage of uphill moves is accepted. After the system has evolved to thermal equilibrium at a given temperature, the temperature is lowered and the annealing process continues until the system reaches a temperature that represents "freezing" ($T = T_{freeze}$). The equilibrium detection at each temperature is a function of the maximum allowable moves at each temperature, N_T, or accept reject limits, N_{acc}/N_T. Thus, simulated annealing combines both iterative improvement in local areas and random jumping to help ensure the system does not get stuck in a local optimum. The general simulated annealing algorithm is given

below (vanLaarhoven and Aarts, 1987).

Initialize variables: T_{init}, T_{freeze}, N_{acc}/N_T, N_T, $S = [\theta]$

While $T \geq T_{freeze}$ do

While $i \leq N_T$ do

Generate move S' by perturbing S

$\Delta cost = cost(S') - cost(S)$

If $\Delta cost \leq 0$ or
(random(0,1) $< e^{-\Delta cost/T}$), then accept S',

$S = S'$

Update N_{acc}/N_T

Until equilibrium is reached at T
(when $i = N_T$)

Update T: $T = \alpha T$

Until $T < T_{freeze}$

A major difficulty in application of simulated annealing is defining the analog to the entities in physical annealing. Specifically, it is necessary to specify the following: the objective function, the configuration space and the move generator, and the temperature schedule. All of the above are dependent on the problem structure. So, for the discrete blending problem we use the following specifications.

Objective. The objective for simulated annealing is identical to the objective given in eq 7, which is to minimize the total mass of frit used over a given combination of blends.

Configuration Space and the Move Generator. Consider the problem where we have the 21 wastes shown in Table 1 (indexed by 1, 2, ..., 21), and we wish to form three blends with these wastes. Our objective is to find the best combination of blends. Suppose an initial state is such that:

Blend 1 = [1, 2, 3, 4, 5, 6, 7]

Blend 2 = [8, 9, 10, 11, 12, 13, 14]

Blend 3 = [15, 16, 17, 18, 19, 20, 21]

A neighbor to this state can be defined as the state which can be reached by the application of a single operator. For a problem with three blends we can devise three simple operators:

1. Swap(1,2), where we swap elements between Blend 1 and Blend 2 ($^1/_3$ probability)
2. Swap(2,3), where we swap elements between Blend 2 and Blend 3 ($^1/_3$ probability)
3. Swap(1,3), where we swap elements between Blend 1 and Blend 3 ($^1/_3$ probability).

We need two more operators to decide which two elements from the two blends are to be swapped. For these studies we have kept an equiprobable chance for one of the seven elements to be chosen from each of the two blends.

Temperatures Schedule.

(1) Initial Temperature. If the initial temperature is too low, the search space is limited and the search becomes trapped in a local region. If the temperature is too high, the algorithm spends a lot of time jumping around wasting CPU time. A rule of thumb for this is to select an initial temperature where a high percentage of moves is accepted. We have chosen 1000 as the initial temperature.

(2) Final Temperature. The final temperature is chosen so that the algorithm terminates after 10 successive temperature decrements with no change in the optimal state.

(3) Temperature Decrement. Choosing the temperature decrement is also rather tricky. If the temperature decrement is too big, the algorithm quickly "quenches" and could get stuck in poor local optima. On the other hand, if the temperature decrement is too small, excessive computational effort is required. A very simple rule which may be used is:

$$T_{new} = \alpha T_{old}$$

where $0.8 \leq \alpha \leq 0.99$. We have chosen α to be 0.95.

Solution. The simulated annealing procedure provided a solution of 11 028 kg of frit (Table 2 provides the frit composition in each blend) which we were able to later confirm to be the global optimum using a branch and bound procedure. The composition of the blends was as follows:

Blend 1

Tanks = [20 3 9 4 8 6 5]

Blend 2

Tanks = [21 12 11 10 19 16 1]

Blend 3

Tanks = [17 15 14 2 18 13 7]

Table 2. Frit Composition in Optimal Solution

component	mass in frit $f^{(i)}$ Blend 1	Blend 2	Blend 3
SiO_2	293.78	680.950	4550.6
B_2O_3	31.350	2.186	1212.4
Na_2O	38.683	375.06	1130.3
Li_2O	43.890	64.709	302.97
CaO	0.000	11.466	344.20
MgO	0.000	66.866	485.78
Fe_2O_3	0.000	0.000	502.11
Al_2O_3	0.000	0.000	640.96
ZrO_2	0.000	0.000	0.000
other	0.000	0.000	250.07

6. A Branch and Bound Procedure

In order to find a guaranteed optimal solution amongst all possible combinations of wastes, each combination must be examined. Consider the example in Figure 1. There is a set of four wastes which has to be partitioned into two blends of two wastes each. Clearly, we have three possible combinations:

[1, 2][3, 4], [1, 3][2, 4], [1, 4][2, 3]

Notice that we are indifferent to the ordering within a blend and also the ordering of blends within a possible combination. That is,

[1, 2][3, 4] ≡ [4, 3][2, 1]

This reduces the number of combinations we need to examine. For each of the three possible blend combina-

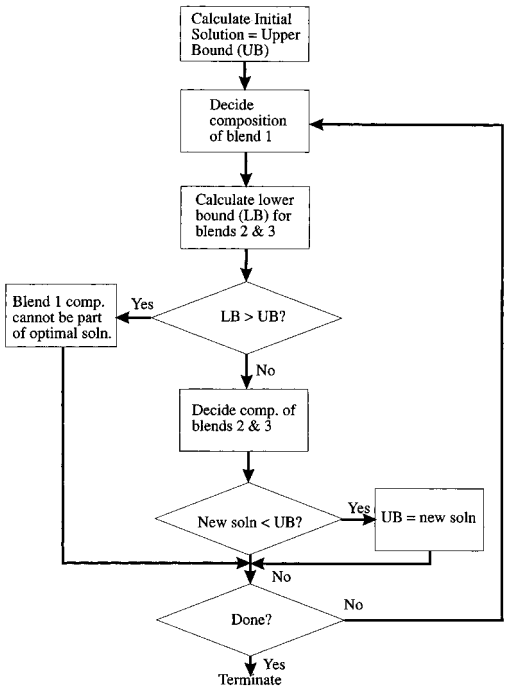

Figure 5. Branch and bound procedure.

tions the amount of frit required for each blend must be found by the NLP. Thus, the enumerative procedure, like simulated annealing, is composed of two procedures. The outer loop is an enumerative procedure which supplies the inner loop the wastes which might be combined to form a blend. In the inner loop the NLP informs the outer loop about the amount of frit necessary to form glass. While this method finds a guaranteed global optimal solution, unfortunately the number of possible combinations we need to examine grows exponentially with the number of wastes available as given in eq 8.

Objective. The objective is to minimize the total amount of frit as given by eq 7.

Bounds. As mentioned before, the test problem with 21 wastes to be partitioned into three blends has 66 512 160 possible combinations to examine. The number of combinations that must be *explicitly* examined to verify optimality can be reduced by using a branch and bound method. The initial configuration is used as the starting *upper bound*. In the case of the test problem, the *lower bound* can be obtained in the following manner:

1. Fix the wastes for the first blend and calculate the amount of frit.

2. Relax the requirement that the remaining wastes must form two blends and assume that they form a single blend. In other words, we remove the binary variables y_{ij} for the two remaining blends. Now calculate the amount of frit required for this relaxation.

3. The total of the frit for the first blend and the relaxation is now a valid lower bound on the original problem.

If the lower bound is greater than the current best upper bound, then any combination where the composition of one of the blends is the same as that of the first blend cannot be optimal. All these combinations can be eliminated and can be considered to be *implicitly* examined. This bounding method was sufficiently strong enough for us to solve the test problem to optimality. However, it still took about 3 days of computation (on a DEC-ALPHA 400 machine) as compared to an average 45 min of CPU time using the two-stage annealing approach.

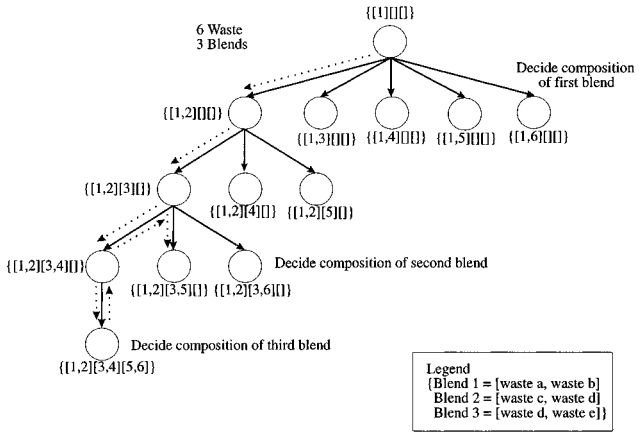

Figure 6. Branch and bound using a depth-first strategy.

Procedure. Figure 5 is a flowchart of the branch and bound procedure for a problem in which three blends are to be formed. An initial solution using any method serves as the initial upper bound. Within a loop which essentially enumerates every possible combination, the procedure first fixes the composition of the first blend. The amount of frit needed for this blend is then determined. By assuming that the remaining two blends form a single blend, the amount of frit for the composite blend is then determined. If the total amount of frit needed for this configuration of blends is greater than the current best solution or the upper bound, then we need not examine any combination of blends where one of the blends is identical to the first blend. But if this is not so, we examine all the possible combinations of the remaining two blends to determine which particular configuration requires the least amount of frit. The upper bound is updated if the best configuration found during enumeration is better than the upper bound. This continues until all possible combinations are either explicitly or implicitly examined. The better the lower bound estimates of the eventual solution, the fewer the amount of explicit enumeration that will have to be performed.

The branch and bound procedure can be implemented using different strategies. We implemented the procedure using a *depth-first* strategy because of its minimal memory requirements. The depth-first strategy is also relatively easy to implement as a recursive function. Any branch and bound method starts off with a start node. With reference to our problem the start node is the assignment of waste 1 to the first blend as shown in Figure 6. Since we are indifferent to the ordering of wastes within a blend and the ordering between blends, it is apparent that this assignment is a valid starting point. In the nodes succeeding the starting node, different wastes which can be combined with waste 1 to form the first blend are considered. As can be seen in Figure 6, there are five nodes which succeed the starting node. If we choose to expand the nodes in the order they are generated, the strategy is called *breadth-first*. If, on the other hand, we choose to expand the most recently generated nodes first, the strategy is called *depth-first*. We see that the depth-first algorithm pushes along one path until it reaches the maximum depth, then it begins to consider alternative paths of the same or less depth that differ only in the last step, then those that differ in the last two steps, etc. As a result of this property the number of nodes which have

to be retained in the memory is very small as compared to the breadth-first algorithm wherein the number of nodes residing in the computer's memory can grow exponentially with the size of the problem. In Figure 6 the dotted arrows indicate the direction in which the search proceeds.

Before going to the third level, a lower bound is computed and compared to the current best solution. If the lower bound is greater than the current best solution, then that part of the tree can be pruned and considered implicitly examined. The search is complete when all branches of the tree are either explicitly or implicitly examined. The better the lower bound is an estimate of the final solution, the fewer the number of branches that have to be explicitly examined.

Optimal Solution. The branch and bound procedure found the optimal solution to be 11 028 kg of frit, which is identical to the solution found by simulated annealing. This confirms that the two-stage SA-NLP approach provided the global optimum with respect to the configuration decisions. As before, the composition of the blends that required the minimum amount of frit was found to be:

Blend 1

Tanks = [20 3 9 4 8 6 5]

Blend 2

Tanks = [21 12 11 10 19 16 1]

Blend 3

Tanks = [17 15 14 2 18 13 7]

7. Conclusions

The purpose of this research was to develop a method which would help to decide which combination of wastes would minimize the amount of frit needed to convert the waste into glass. The benefit of reducing the amount of frit used is in reduced material costs and the reduced bulk of the glass formed which, in turn, reduces the disposal costs. The search space grows exponentially with an increase in parameters defining the problem, making it almost impossible to find optimal solutions for realistically sized problems. We compared different combinatorial optimization techniques such as GAMS-based MINLP algorithm, branch and bound, and simulated annealing and heuristics approaches to find the optimal solution. We have found that a two-stage approach combining simulated annealing and NLP algorithms is a cost-effective means to obtain global optimal or near global optimal solutions with reasonable amounts of computational effort. Both the heuristic approach and GAMS-based MINLP result in a local minimum. The branch and bound procedure leads to a global optimum but requires significantly longer computational time than the coupled simulated annealing-NLP approach.

Appendix. Details of Glass Property Constraints

Notation

C_1 = bound for Crystal1 = 3.0
C_2 = bound for Crystal2 = 0.08
C_3 = bound for Crystal3 = 0.225
C_4 = bound for Crystal4 = 0.18
C_5 = bound for Crystal5 = 0.18
k_{min} = lower limit for conductivity = 18
k_{max} = upper limit for conductivity = 50
μ_{min} = lower limit for viscosity (PaS) = 2.0
μ_{max} = upper limit for viscosity (PaS) = 10.0
D_{max}^{PCT} = max release rate (product consistency test) (g/m^2) = 10.0
D_{max}^{MCC} = max release rate (materials characterization center) (g/m^2) = 28.0
μ_a^i = linear coefficients of viscosity model
μ_b^{ij} = cross-term coefficients of viscosity model
k_a^i = linear coefficients of electrical conductivity model
k_b^{ii} = cross-term coefficients of electrical conductivity model
Dp_a^i = linear coefficients of durability (PCT) model (for boron)
Dp_b^{ij} = cross-term coefficients of durability (PCT) model (for boron)
Dm_a^i = linear coefficients of durability (MCC) model (for boron)
Dm_b^{ij} = cross-term coefficients of durability (MCC) model (for boron)

Bounds

1. Component Bounds:

(a) $0.42 \le p^{(SiO_2)} \le 0.57$
(b) $0.05 \le p^{(B_2O_3)} \le 0.20$
(c) $0.05 \le p^{(Na_2O)} \le 0.20$
(d) $0.01 \le p^{(Li_2O)} \le 0.07$
(e) $0.0 \le p^{(CaO)} \le 0.10$
(f) $0.0 \le p^{(MgO)} \le 0.08$
(g) $0.02 \le p^{(Fe_2O_3)} \le 0.15$
(h) $0.0 \le p^{(Al_2O_3)} \le 0.15$
(i) $0.0 \le p^{(ZrO_2)} \le 0.13$
(j) $0.01 \le p^{(other)} \le 0.10$

2. Five Glass Crystallinity Constraints:

(a) $p^{(SiO_2)} > p^{(Al_2O_3)} C_1$
(b) $p^{(MgO)} + p^{(CaO)} < C_2$
(c) $p^{(Fe_2O_3)} + p^{(Al_2O_3)} + p^{(ZrO_2)} + p^{('Other')} < C_3$
(d) $p^{(Al_2O_3)} + p^{(ZrO_2)} < C_4$
(e) $p^{(MgO)} + p^{(CaO)} + p^{(ZrO_2)} < C_5$

3. Solubility Constraints:

(a) $p^{(Cr_2O_3)} < 0.005$
(b) $p^{(F)} < 0.017$
(c) $p^{(P_2O_5)} < 0.01$
(d) $p^{(SO_3)} < 0.005$
(e) $p^{(Rh_2O_3+PdO+Ru_2O_3)} < 0.025$

4. Viscosity Constraints:

$$\text{(a)} \sum_{i=1}^{n} \mu_{\mathrm{a}}^{i} p^{(i)} + \sum_{j=1}^{n} \sum_{i=1}^{n} \mu_{\mathrm{b}}^{ij} p^{(i)} p^{(j)} > \log(\mu_{\min})$$

$$\text{(b)} \sum_{i=1}^{n} \mu_{\mathrm{a}}^{i} p^{(i)} + \sum_{j=1}^{n} \sum_{i=1}^{n} \mu_{\mathrm{b}}^{ij} p^{(i)} p^{(j)} < \log(\mu_{\max})$$

5. Conductivity Constraints:

$$\text{(a)} \sum_{i=1}^{n} k_{\mathrm{a}}^{i} p^{(i)} + \sum_{j=1}^{n} \sum_{i=1}^{n} k_{\mathrm{b}}^{ij} p^{(i)} p^{(j)} > \log(k_{\min})$$

$$\text{(b)} \sum_{i=1}^{n} k_{\mathrm{a}}^{i} p^{(i)} + \sum_{j=1}^{n} \sum_{i=1}^{n} k_{\mathrm{b}}^{ij} p^{(i)} p^{(j)} < \log(k_{\max})$$

6. Dissolution Rate for Boron by PCT Test (DissPCTbor):

$$\sum_{i=1}^{n} Dp_{\mathrm{a}}^{i} p^{i} + \sum_{j=1}^{n} \sum_{i=1}^{n} Dp_{b}^{ij} p^{(i)} p^{(j)} < \log(D_{\max}^{\mathrm{PCT}})$$

7. Dissolution Rate for Boron by MCC Test (DissMCCbor):

$$\sum_{i=1}^{n} Dm_{\mathrm{a}}^{i} p^{i} + \sum_{j=1}^{n} \sum_{i=1}^{n} Dm_{\mathrm{b}}^{ij} p^{(i)} p^{(j)} < \log(D_{\max}^{\mathrm{MCC}})$$

Literature Cited

Beale, E. M. *Integer Programming: The State of the Art in Numerical Analysis*; Academic Press: London, 1977.

Benders, J. F. Partitioning for solving mixed-variables programming problems. *Numer. Math.* **1962**, *4*, 238.

Biegler, L. T. Simultaneous Modular Simulation and Optimization. Proceedings of Second International Conference on Foundations of Computer Aided Process Design, Snowmass, CO 1983.

Biegler, L. T.; Cuthrell, J. E. Improved Infeasible Path Optimization for Sequential Modular Simulators. Part II. *Comput. Chem. Eng.* **1985**, *9*, 257.

Brooke, A.; Kendrick, D.; Meeraus, A. *GAMS User's Guide, Release 2.25*; The Scientific Press: San Francisco, CA, 1992.

Collins, N. E.; Eglese, R. W.; Golden, B. L. Simulated Annealing—an annotated bibliography. *Am. J. Math. Mgmt. Sci.* **1988**, *8*, 209.

Diwekar, U. M.; Grossmann, I. E.; Rubin, E. S. An MINLP process synthesizer for a sequential modular simulator. *Ind. Eng. Chem. Res.* **1992**, *31*, 313.

Duran, M. A.; Grossmann, I. E. An outer-approximation algorithm for a class of Mixed-Integer Nonlinear Programs. *Math. Program.* **1986**, *36*, 307.

Edgar, T. F.; Himmelblau, D. M. *Optimization of Chemical Processes*; McGraw-Hill Book Co.: New York, 1988.

Floudas, C. S.; Aggrawal, A.; Ciric, A. R. A global optimum search for nonconvex NLP and MINLP problems. *Comput. Chem. Eng.* **1989**, *13*, 1117.

Geoffrion, A. M. Generalized Benders decomposition. *J. Optim. Theory Appl.* **1972**, *10*, 237.

Gill, P. E.; Murray, W.; Wright, M. H. *Practical Optimization*; Academic Press: London, 1981.

Gill, P. E.; Murray, W.; Saunders, M. A.; Wright, M. H. Constrained Nonlinear Programming. Technical Report SOL 87-13; Stanford University: Stanford, CA, 1987.

Gupta, O. K. *Branch and bound experiments in nonlinear integer programming.* Ph.D. Thesis, Purdue University, West Lafayette, IN, 1980.

Haverly, C. A. Studies of the behavior of recursion for the pooling problem. *SIGMAP Bull.* **1978**, *25*, 19.

Hoza, M. Multipurpose optimization models for high-level waste vitrification. *Proceedings of the International Topical Meeting on Nuclear and Hazardous Waste Management—SPECTRUM '94*; American Nuclear Society: La Grange Park, IL, 1994; pp 1072−1077.

Kocis, G. R.; Grossmann, I. E. Relaxation strategy for the structural optimization of process flowsheets. *Ind. Eng. Chem. Res.* **1987**, *26*, 1869.

Lasdon, L. S.; Warren, A. D. Survey of nonlinear programming applications, **1980**, *28* 1029.

Morton, T. E.; Pentico, D. *Heuristic Scheduling Systems—With Applications to Project Scheduling*; John Wiley: New York, 1993.

Palacios-Gomez, F. E. The solution of nonlinear optimization problems using successive linear programming system. Ph.D. Dissertation, School Business Administration, The University of Texas at Austin, TX, 1980.

Reklaitis, G. V.; Ravindran, A.; Ragsdell, K. M. *Engineering Optimization*; John Wiley & Sons, Inc.: New York, 1983.

Rich, E.; Knight, K. *Artificial Intelligence*; McGraw-Hill: New York, 1991.

Taha, H. A. *Integer programming: theory, applications, and computations*; Academic Press: New York, 1975.

Thomas, G. S.; Jenning, J. C.; Abbott, P. A blending problem using integer programming on-line. *Math. Program. Stud.* **1978**, *9*, 30.

vanLaarhoven, P. J. M.; Aarts, E. H. L. *Simulated Annealing: Theory and Applications*; Reidel Publishing Co.: Amsterdam, Holland, 1987.

Viswanathan, J.; Grossmann, I. E. A combined penalty function and outer-approximation method for MINLP optimization. *Comput. Chem. Eng.* **1990**, *14*, 769.

Received for review January 17, 1996
Revised manuscript received May 14, 1996
Accepted May 15, 1996[⊗]

IE960028C

[⊗] Abstract published in *Advance ACS Abstracts,* September 15, 1996.