Derived Heuristics Based Consistent Optimization of Material Flow in a Gold Processing Plant

Christie Myburgh
Principal Research & Development Engineer
Maptek Private Limited
Northbridge, WA 6003, Australia
Email: christie.myburgh@maptek.com.au

Kalyanmoy Deb
Professor, Dept. of Elect. and Comp. Engg.
Michigan State University
East Lansing, MI 48824, USA
Email: kdeb@egr.msu.edu

COIN Report Number 2016025

Abstract

Material flow in a chemical processing plant often follows complicated control laws and involve plant capacity constraints. Importantly, the process involves discrete scenarios which when modeled in a programming format involves if-then-else statements. Therefore, a formulation of an optimization problem of such processes becomes complicated with nonlinear and non-differentiable objective and constraint functions. In handling such problems using classical point-based approaches, users often have to resort to fix-ups and indirect ways of representing the problem to suit the restrictions associated with classical methods. In a particular gold processing plant optimization problem, we demonstrate these facts by showing results from MATLAB's well-known fmincon routine. Thereafter, we develop a customized evolutionary optimization procedure which is capable of handling all complexities offered by the problem. Although the evolutionary approach produced results with comparatively less variance over multiple runs, the performance has been enhanced by introducing derived heuristics associated with the problem. In this paper, we demonstrate the development and usage of derived heuristics in a practical problem and show their importance in a quick convergence of the overall algorithm.

1 Introduction

Optimization problems come in different forms and complexities from practice. Except a very few optimization problems having linear or convex objective and constraint functions, there is no optimization algorithm that comes with a mathematical proof of finding the exact optimal solution starting from any arbitrary initial solution in a finite computational effort to solve an arbitrary problem [3, 5, 17]. When problems are nonlinear, non-differentiable and deviate from the requirements of provable algorithms, any algorithm is as good or bad as another, if an algorithm needs to be used to solve a large set of problems [20, 4]. It is clear that in solving such problems, problem heuristics must be embedded in an optimization algorithm for a computationally quick and consistent application [9]. In certain problems, problem heuristics may be available a priori and before the application of an optimization algorithm. It then makes sense to use available problem heuistics to *customize* an algorithm. See for example, a recent study by the authors where a billion variables are handled using a customized population-based algorithm [9]. On the other hand, important and useful heuristics may not be always available in an optimization problem before the optimization task is undertaken, but what we portray in this paper is that problem heuristics can be *derived* from intermediate solutions obtained either during a systematic

optimization process or after some preliminary optimization runs are executed. We propose and illustrate such a derived heuristics based customized optimization procedure in this paper through a gold processing plant optimization problem.

In this paper, we demonstrate the use of *derived heuristics* which are obtained from multiple applications of an optimization algorithm on various input settings of the original problem. The derived heuristics are then embedded within an optimization algorithm so that the algorithm is able to enforce derived heuristics towards achieving a more consitent and reliable performance in a computationally fast and routine manner. It is important to highlight that the derived heuristics are not only useful from an optimization point of view, they also remain as valuable problem knowledge to the user. We also highlight here the fact that embedding of heuristics of an arbitrary structure in an optimization algorithm requires the algorithm to be amenable to such changes. We argue and demonstrate here the suitability of population-based evolutionary optimization methods [11, 5] vis-a-vis point-based classical optimization methods [17, 16] in achieving such a customization task.

To demonstrate the use of derived heuristics in enhancing performance of an optimization algorithm, we consider a gold processing plant optimization problem. In the recent past, a surge of applications have emerged in solving gold and other chemical extraction process optimization problems [13, 19, 1, 10, 12, 14, 18, 15, 2]. The gold extraction plant involves several processing options depending on quality, size, and lithological composition of incoming material to the plant. The operating principles of each process within the plant is involved and obtained by careful individual chemical analysis in laboratories. The resulting optimization problem involves discrete and continuous variables and the formulated objective and constraint functions are far from being linear or convex. Due to the use of 'if-then-else' statements and minimum and maximum operators (which introduces non-differentiability) in deriving objective and constraint functions, the formulation is procedural. Any idea of the structure (convex or otherwise) of these functions is difficult to comprehend from their descriptions. Thus, it becomes difficult to decipher any useful problem heuristics from the problem. Moreover, all these practicalities cause the use of a point-based deterministic classical optimization algorithm difficult. However, in this study, we have used a fix-up to suit its solution using MATLAB's fmincon routine – a software popularly used by optimization practitioners. Although resonably good results are obtained, they fail to produce consistently good results over multiple runs, each starting from a different initial guess solution.

Next, we modify an existing real-parameter genetic algorithm (RGA) to solve the same gold processing optimization problem in its totality and without any fix-ups. Results demonstrate a more consistent performance of the proposed RGA. We then analyze obtained RGA results and decipher the optimized solutions to discover properties of the optimized solutions. For example, we observe that to achieve the optimal performance, one of the mills associated with the processing plant must be operated at its full capacity. Such a vital information was not obvious before execting the optimization task. Once this information is revealed from optimized solutions of multiple instantiations of the problem, we then use it to modify the RGA procedure so that every solution satisfies the specific capacity constraint directly. We argue and demonstrate in this paper that an use of such derived heutistics enhances the performance of an optimization algorithm and is able to produce more consistent solutions and in a computationally faster manner than the original algorithm.

In the remainder of the paper, we provide a problem formulation of the gold processing optimization problem in Section 2. In Section 3, we describe the proposed modified RGA procedure for solving the gold processing optimization problem. Thereafter, in Section 4, we consider one instance of the problem and apply MATLAB's fmincon routine. Results are then described. This section also represents results from modified RGA, which are then compared with fmincon re-

sults. Section 5 proposes the derived heuristics based optimization procedure and presents results. Finally, conclusions of the study are made in Section 6.

2 Problem Definition

A specific gold extraction process (shown in Figure 1) applies several operations on different lithological materials mixed with gold (AU) and sulphur (S). Let us say that there are n lithological materials supplied at the inlet of the extraction process and the amount (in ton) of each lithology is available. Say these amounts are W_i ton for i-th lithology. The first and foremost important decision that needs to be taken is the proportion of these amounts to be fed to two Mills (Mill 1 and Mill 2). Mill 2 processes the materials directly and sends the upgraded milled materials to an autoclave (AC), whereas the materials from Mill 1 reaches the autoclave via a Beneficiary process. Let us denote the proportion of W_i to be fed to Mill 1 as x_i and remaining proportion ($y_i = 1 - x_i$) is fed to Mill 2, as shown in the adjoining figure.

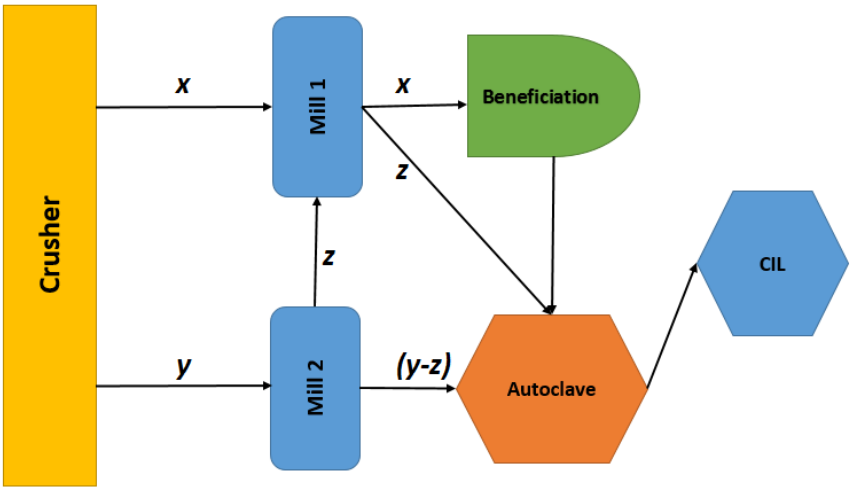

Figure 1: A sketch of the gold processing plant layout.

Both mills can crush the materials into one of K sizes ($s = 1, 2, \ldots, K$), each requiring different power levels (p_i^s, $i = 1, 2, \ldots, n$, $s = 1, 2, \ldots, K$). Both Mill 1 and Mill 2 have a capacity limit (T_1 and T_2, respectively) and also have their own maximum power handling capacities (P_1 and P_2, respectively). If all the materials fed to Mill 2 cannot be processed either due to maximum capacity or power requirements, the excess lithological materials ($(z_1, z_2, \ldots, z_n)$ are sent to Mill 1. These excess materials are then grinded in Mill 1 and supplied directly to the autoclave. The flow of these excess materials is shown in the figure with lines marked with **z**.

The autoclave oxidizes the incoming material and attempts to remove sulphur from gold. The autoclave also has a maximum capacity. T_{AC}. The task of the optimization problem is to maximize the amount of gold metal at the Carbon-In-Leach (CIL) plant. The beneficiary plant also has a maximum capacity T_B. Each of the process is involved and we attempt to formulate the optimization problem in the following paragraphs.

The variables of the optimization problem are **x** and **z**-vectors of size n each. Note that each of these variables must be non-negative. Thus, the amounts of different lithological materials processed at Mill 2 are $y_i' = y_i - z_i$ for $i = 1, 2, \ldots, n$. The capacity and power constraints can be

written as follows:

$$g_1(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n}(x_i+z_i)W_i - T_1 \leq 0, \quad (1)$$

$$g_2(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n}(y_i-z_i)W_i - T_2 \leq 0, \quad (2)$$

$$g_3^s(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n}(x_i+z_i)W_i p_i^s - P_1 \leq 0, \quad \text{for } s=1,2,\ldots,K, \quad (3)$$

$$g_4^s(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n}(y_i-z_i)W_i p_i^s - P_2 \leq 0, \quad \text{for } s=1,2,\ldots,K, \quad (4)$$

$$g_5(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n} x_i W_i - T_B \leq 0. \quad (5)$$

To formulate the capacity constraint of the autoclave, we need to compute thermal limit $\mathcal{H}$ given as follows:

$$\mathcal{H}^s = \zeta\left(\frac{\beta \bar{S}_{net}^s}{B_T + D_T} - \gamma\right),$$

where $\bar{S}_{net}^s = B_T B_S^s + D_T D_S$ and ζ, β, and γ are constants. The Beneficiary tonnage is given as follows:

$$B_T = f_{my}\sum_{i=1}^{n} x_i W_i,$$

where f_{my} is the mass recovery factor. The Beneficiary sulphur B_s is defined as follows:

$$B_s^s = \frac{G_S \bar{\alpha}_S^s}{100 f_{my}},$$

where G_S is the incoming sulphur grade to beneficiary and is given as follows:

$$G_S = \frac{\sum_{i=1}^{n} S_i(x_i+z_i)W_i}{\sum_{i=1}^{n}(x_i+z_i)W_i},$$

where S_i is the sulphur grade in original material of each i-th lithology. The updated beneficiary sulphur recovery α_S' is given as follows:

$$\bar{\alpha}_S^s = \alpha_S - \sum_{i=1}^{n} R_i - es + h,$$

The parameters e and h are user-defined constants which depend on the grind size s. Here α_S is the beneficiary sulphur recovery and is given as follows:

$$\alpha_S = a + b f_{my} - c G_S,$$

where a, b, and c are user-supplied constants. The penalty R_i comes from specific lithologies that degrade the efficiency of the beneficiation process and is defined as follows:

$$R_i = d_i \frac{x_i W_i}{\sum_{i=1}^{n} x_i W_i},$$

where d_i is a constant for i-th lithography. The parameters e and h are constants and s is the grind size discussed before.

The direct tonnage D_T and direct sulphur grade D_S are given as follows:

$$\begin{aligned} D_T &= \sum_{i=1}^{n} y_i W_i, \\ D_S &= \frac{G_S \sum_{i=1}^{n} z_i W_i + \sum_{i=1}^{n} S_i (y_i - z_i) W_i}{\sum_{i=1}^{N} y_i W_i}. \end{aligned}$$

For autoclave capacity calculation, we now compute the throughput mass, as follows:

$$\eta^s = \min\left(\mathcal{H}^s, T_{AC}, (B_T + D_T)\right).$$

The capacity constraint can now be written as follows:

$$g_6^s(\mathbf{x}, \mathbf{z}) = (B_T + D_T) - \eta^s \leq 0. \tag{6}$$

This ensures that the autoclave can process all the incoming materials.

Now, we discuss the procedure for computing the objective function, which is the gold metal ounces (f) produced by the plant, defined as the product of gold recovery AU_y and net gold in ounces AU_{net}:

$$f^s(\mathbf{x}, \mathbf{z}) = \frac{AU_y AU_{net}^s}{31.1035}, \tag{7}$$

The division by 31.1035 converts the grams to ounces. The net gold available after the autoclave AU_{net} comes from both beneficiary process and direct feed, as follows:

$$AU_{net}^s = D_{AU} D_T + B_{AU}^s B_T.$$

The direct gold amount D_{AU} is calculated similar to direct sulphur amount computation done before:

$$D_{AU} = \frac{G_{AU} \sum_{i=1}^{n} z_i W_i + \sum_{i=1}^{n} AU_i (y_i - z_i) W_i}{\sum_{i=1}^{N} y_i W_i},$$

where gold grade G_{AU} is given below:

$$G_{AU} = \frac{\sum_{i=1}^{n} AU_i (x_i + z_i) W_i}{\sum_{i=1}^{n} (x_i + z_i) W_i}.$$

The beneficiary gold grade B_{AU} is calculated like B_S, as follows:

$$B_{AU}^s = \frac{G_{AU} \bar{\alpha}_{AU}^s \sum_{i=1}^{n} x_i W_i}{f_{my}},$$

where updated gold beneficiary recovery $\bar{\alpha}_{AU}^s$ is calculated as follows:

$$\bar{\alpha}_{AU}^s = \alpha_{AU} - \sum_{i=1}^{n} Q_i - ms + o,$$

The parameters m and o are user-defined constants which depend on the grind size s. The gold beneficiary recovery α_{AU} is given as follows:

$$\alpha_{AU} = q + r f_{my} + t G_{AU} - u G_S.$$

The penalty Q_i is given as $Q_i = v_i \frac{x_i W_i}{\sum_{i=1}^{n} x_i W_i}$, where v_i is a constant set for i-th lithography. The parameters q, r, t and u are constants.

The computation of gold recovery AU_y is discussed next. This requires calculation of Oxidation, given as follows:

$$\text{Oxidation}^s = \frac{100\phi}{\psi \bar{S}^s_{net}},$$

where ϕ and ψ are user-defined constants. If Oxidation is greater than $\text{OX}_{\max}$, it is set to $\text{OX}_{\max}$. Then, AU_y is defined as follows:

$$AU_y = \begin{cases} \kappa\,(\text{Oxidation}^s) + \mu, & \text{if Oxidation}^s \geq \text{OX}_{\min}, \\ \tau, & \text{otherwise.} \end{cases} \tag{8}$$

There is one more constraint regarding the oxidation of gold, making sure that the oxidation level is above the minimum level:

$$g^s_7(\mathbf{x}, \mathbf{z}) = \text{OX}_{\min} - \text{Oxidation}^s \leq 0. \tag{9}$$

The above description indicates that the problem has both $\mathbf{x}$ and $\mathbf{z}$-vectors as variables. But to save computational time, we follow a simple ordering strategy to determine $\mathbf{z}$-vector from $\mathbf{x}$-vector. For a given $\mathbf{x}$-vector violating either the capacity or power constraints of Mill 2, we use one of the following two ordering strategies to determine the $\mathbf{z}$-vector.

1. Lithologies are sorted in increasing order of the gold grade (AU_i). We call this event with an index value of $j = 1$.

2. Lithologies are sorted in decreasing order of the gold grade (AU_i). We call this event with an index value of $j = 2$.

After any of the above sorting is done, the lithologies ($\mathbf{y}$) are kept for Mill 2 starting from the top of the sorted list until the capacity or power constraints are exceeded. The remaining materials ($\mathbf{z}$) are sent to Mill 1. Thus, $\mathbf{z}$-vector is computed from the given $\mathbf{x}$-vector and by using some of the resource constraints. The above two orderings are performed separately and the best scenario is chosen.

The objective function f^s and three constraint functions g^s_3, g^s_4 and g^s_6 needs to be computed for each specified grind size s. The above three constraints must be satisfied for all grind sizes, while the maximum value of objective function computed for all grind sizes needs to be maximized. Since the ordering process described above needs to be performed separately, each solution $\mathbf{x}$ needs to computed for constraint and objective function a total of $2K$ times. Thus, for a variable vector ($\mathbf{x}$, $\mathbf{z}$), we identify all (s, j) combinations for which *all* constraints are satisfied. Let us say that the indices set for such combinations is (I_s, I_j). Here, we provide the final formulation of the optimization problem:

$$\begin{array}{ll} \text{Maximize} & \max_{(s,j)\in(I_s,I_j)} f^{s,j}(\mathbf{x}, \mathbf{z}), \\ \text{subject to} & g^j_1(\mathbf{x}, \mathbf{z}) \leq 0, \quad \text{for } j \in I_j, \\ & g^j_2(\mathbf{x}, \mathbf{z}) \leq 0, \quad \text{for } j \in I_j, \\ & g^{s,j}_3(\mathbf{x}, \mathbf{z}) \leq 0, \quad \text{for } (s, j) \in (I_s, I_j), \\ & g^{s,j}_4(\mathbf{x}, \mathbf{z}) \leq 0, \quad \text{for } (s, j) \in (I_s, I_j), \\ & g^j_5(\mathbf{x}, \mathbf{z}) \leq 0, \quad \text{for } j \in I_j, \\ & g^{s,j}_6(\mathbf{x}, \mathbf{z}) \leq 0, \quad \text{for } j \in I_j, \\ & g^{s,j}_7(\mathbf{x}, \mathbf{z}) \leq 0, \quad \text{for } j \in I_j, \\ & 0 \leq x_i \leq 1, \quad 0 \leq z_i \leq 1, \quad \text{for } i = 1, 2, \ldots, n. \end{array} \tag{10}$$

3 Proposed Optimization Algorithm

The description of the problem makes it clear that it is a nonlinear problem involving different complexities, such as if-then clause, non-differentiable objective function due to the max-operator, multiple case iterations. These are the reasons we choose a population-based and non-gradient optimization method.

3.1 Initialization Procedure

We randomly create a $\mathbf{x}$-vector and then apply a repair operator on it. The repair operator ensures that the beneficiary capacity constraint (g_5) is satisfied. If $g_5(\mathbf{x}) > 0$ (violation), then we proportionately reduce each x_i so that g_5 becomes active:

$$x_i \leftarrow (x_i W_i) \frac{T_B}{\sum_{i=1}^{n} x_i W_i}. \tag{11}$$

This process reduces consideration of one constraint in the evaluation process.

3.2 Evaluation of a Solution

A population member $\mathbf{x}$ is evaluated as follows. As it is discussed above, we do not need to check constraint g_5 for violation. Next, for each combination of size (s) and ordering (j), we accept lithologies (y_i) for Mill 2 until we hit either the capacity or the power constraint, whichever comes first. The remaining lithologies are then redirected to Mill 1. This process is simulated as follows. If a lithology is completely accepted, the respective z_i is equal to zero. For other lithologies which are either not accepted at all or partially, the respective z_i is set. At the end of this process, $\mathbf{z}$ is completely known.

While redirecting the lithologies to Mill 1, constraints g_1 and g_3 are checked simultaneously. If at any point, the total material ($\sum_{i=1}^{n}(x_i + z_i)W_i$) or power required $\sum_{i=1}^{n}(x_i + z_i)W_i p_i^s$ exceeds the Mill 1 capacity constraint (g_1) or Mill 1 power capacity constraint (g_3), then the remaining z_i's are set to zero, meaning that no further material can be accepted by Mill 1. This process automatically satisfies constraints g_1, g_2 and g_4.

Next, we compute constraints g_6 and g_7 and check for their feasibility. If any of them are violated, an overall constraint violation is computed as follows:

$$CV(\mathbf{x}) = \frac{\langle g_6^{s,j}(\mathbf{x})\rangle}{\eta^s} + \frac{\langle g_7^{s,j}(\mathbf{x})\rangle}{\mathrm{OX}_{\min}}, \tag{12}$$

where the bracket operator $\langle\alpha\rangle$ is equal to α if $\alpha > 0$; zero, otherwise.

We compute the objective function for all combinations of (s, j) such that $CV(\mathbf{x}) = 0$ and declare the maximum value as the objective function value. If for a solution $\mathbf{x}$, no combination of (s, j) makes all constraints satisfied, we choose the combination having minimum $CV(\mathbf{x})$ and set the respective objective value $f^s(\mathbf{x}) = 0$. Thus, at the end of the evaluation process, we shall have two quantities for each solution: $f^s(\mathbf{x})$ and $CV(\mathbf{x})$. If $CV(\mathbf{x}) = 0$, then the solution $\mathbf{x}$ is feasible, otherwise it is not.

3.3 Selection Operator

Two population members are randomly chosen [7] and then the better one is chosen [6]. If one of the two solutions is feasible and other is not, the feasible one is chosen. But if both are feasible, the one with larger objective function value is chosen. In the third case when both variables are

infeasible, the one with smaller overall constraint violation (function CV defined in Equation 12) is chosen. The selected individuals are saved in the mating pool for recombination and mutation operations, describe below.

3.4 Recombination Operator

We use the simulated binary crossover (SBX) [8] operator with $p_c = 0.92$ and $\eta_c = 2$ on a pair of individuals from the mating pool. Every parent pair is checked with a probability p_c for recombination. If the pair is to be recombined, two values for each variable are then recombined one at a time with 50% probability to create two new offspring values. In case of no recombination, the parent values are retained. After all variables are recombined this way, they are then concatenated to create offspring solution vectors.

3.5 Mutation Operator

Polynomial mutation operator [7] is used next to create a neighboring solution by using $p_m = 0.01$ and $\eta_m = 2$ to every solution created by the recombination operator. First, every variable is checked with a probability p_m for mutation. If successful, the variable is mutated to a neighboring value by using a polynomial probability function with an order η_m.

3.6 Repair Operator

Both new recombined and mutated offspring solutions are then checked for any violation of constraint g_5 and $\mathbf{x}$-vector is adjusted accordingly, as described in Equation 11.

3.7 Termination Condition

The variation of the objective function value for the population-best solution is checked over the past 30 generations. If the variation is less than 0.05%, the optimization run is terminated. There is also an upper limit of 400 generations for termination, as otherwise it would require a large computational burden.

4 Results

For a particular gold processing plant, we gather the following parameter values which are used in this study. There are five ($n = 5$) lithological units in the problem.

$a = 95.75$,	$b = 12.29$,	$f_{my} = 0.3$,	$c = 1.525$,
$d = (2.6, 2.6, 1.3, 1.3, 0)$,	$e = 0.0775$,	$h = 8.4189$,	$m = 0.0584$
$o = 6.3247$,	$q = 78.74$,	$r = 0.2811$	$t = 1.789$
$u = 1.84$,	$\mathrm{OX}_{\min}$=0.55,	$q = (4, 4, 4, 4, 0)$	$\mathrm{OX}_{\max} = 0.94$
$T_1 = 15$ M-Ton,	$T_2 = 4.7$ M-Ton,	$P_1 = 162.936$ GW/yr,	$P_2 = 73,584$ GW/yr
$T_B = 9$ M-Ton,	$T_{AC} = 11$ M-Ton,	$\beta = 341$,	$\zeta = 8760$
$\gamma = 649$,	$\phi = 1.314$ M-Ton,	$\psi = 1.87$,	$\kappa = 0.21$
$\mu = 0.716$,	$\tau = 0.3$		

The power p_i^s (in kW/hr/Ton) for each lithology (i) and size (s) are shown in the table. Note that there are $K = 6$ grind sizes in this problem.

Lith.	Grind size, s (μ m)					
i	106	125	150	180	210	250
1	14.7	13.7	12.7	11.7	11.0	10.2
2	10.0	9.5	8.9	8.4	8.0	7.5
3	18.5	17.4	16.2	15.1	14.2	13.3
4	17.8	16.7	15.6	14.5	13.7	12.7
5	18.4	17.3	16.1	15.1	14.2	13.3

Our first problem has the following lithological composition (net weight, gold content and sulphur content):

$$\begin{aligned} \mathbf{W} &= (248,083, \quad 48,350, \quad 121,824, \quad 1,042,726, \quad 11,561,044) \text{ Ton} \\ \mathbf{AU} &= (2.68, \quad 2.91, \quad 3.84, \quad 3.30, \quad 2.60) \text{ gm/Ton} \\ \mathbf{S} &= (5.9, \quad 6.15, \quad 3.32, \quad 5.25, \quad 3.80) \ \% \end{aligned}$$

4.1 Results Using MATLAB's fmincon

In order to investigate how a classical optimization algorithm will solve the above problem, we have coded the above optimization problem into MATLAB's fmincon routine, which incorporates latest methods and is popularly used.

To implement the problem as it is described above, there are two potential difficulties for directly providing objective and constraint value information according to the required (and often rigid) structure of a standard software, which we discuss first.

1. The first difficulty arises due to the fact that the optimization problem involves evaluating the constraints for K different grind sizes and then evaluating the maximum objective value from all grind sizes that make all constraints satisfied and then supplying the maximum of all such feasible objective values. Moreover, after the particular grind size is identified for any solution, the constraint values have to be supplied for that specific grind size. This requires computation of constraints within both constraint and objective function modules of MATLAB. To avoid such complications, we have individually run the complete optimization problem for each grind size and then report the results for the best grind size. It is worth mentioning here that such practical complexities in an optimization problem description are often negotiated by a simple fix-up strategy, which although allows a point-based and often rigidly-structured optimization algorithm to be applied to the problem, but often it comes at the expense of more computational time and solution evaluations.

2. The second difficulty arises in determining the exact composition of the excess materials that need to be sent from Mill 2 to Mill 1 due to power or capacity limit violation of Mill 2. Since determination of such quantities is often procedural and difficult to adopt in a structured software, we consider the amount of excess materials as an additional variable vector ($\mathbf{z}$). Thus, the overall problem involves twice the number of variables than the original problem ($\mathbf{x}$, $\mathbf{z}$).

We now present the results obtained using MATLAB's fmincon routine. To investigate the variations in optimized objective values over multiple runs, we ran fmincon routine 11 times from different initial points, created at random, each within $[0,1]^{2n}$. The first n variables are directly used as $\mathbf{x}$, while the second set of n variables signify the proportion of $\mathbf{y}$ $(= 1 - \mathbf{x})$ that should be declared as excess materials of Mill 2.

Table 1 presents the best, median and worst objective value and the respective solution ($\mathbf{x}$, $\mathbf{z}$) obtained by the routine for the input data of the problem described at the start of this section.

Table 1: Results of 11 runs of fmincon routine are shown.

Variable-vector **x**					Variable-vector **z**					Grind	Objective	Func.
x_1	x_2	x_3	x_4	x_5	z_1	z_2	z_3	z_4	z_5	(μ-m)	(Oz)	Evals.
0.133	0.001	0.000	0.000	0.409	0.002	0.000	1.000	0.000	0.453	150	956,708.73	5,980
0.000	0.000	0.001	0.000	0.419	0.000	0.000	0.999	0.000	0.502	180	949,333.64	6,469
0.040	0.021	0.002	0.011	0.406	0.960	0.000	0.248	0.002	0.380	125	962,535.32	6,427
0.000	0.789	0.000	0.603	0.351	0.000	0.211	0.994	0.077	0.362	125	958,894.19	7,730
0.000	0.000	0.000	0.000	0.412	0.000	0.001	0.948	0.066	0.294	150	956,643.12	4,714
0.034	0.000	0.244	0.136	0.392	0.000	0.195	0.755	0.000	0.334	125	961,977.89	5,385
0.000	1.000	0.011	0.044	0.407	0.691	0.000	0.988	0.939	0.205	150	952,870.45	8,770
1.000	1.000	0.000	0.000	0.383	0.000	0.000	1.000	0.152	0.376	125	961,003.88	5,992
0.000	0.000	1.000	0.000	0.397	1.000	1.000	0.000	0.003	0.334	125	961,935.82	10,736
0.606	0.000	0.000	1.000	0.320	0.000	1.000	0.000	0.000	0.295	180	941,499.67	6,800
0.000	0.000	0.000	0.000	0.407	0.000	0.001	0.000	0.000	0.403	125	963,189.57	2,349

The above table shows that there is a considerable variation in optimized objective value in all 11 independent runs. The best, median, average, and worst optimized objective values are 963,189.58, 958,894.19, 956,962.93, and 941,499.67 ounces, respectively. The best, median, average and worst number of function evaluations are 2,349, 6,427, 6,486.5, and 10,736, respectively. Over 11 runs, there is 2.27% variation in optimized objective value and 129.30% variation in number of function evaluations. Also notice the wide variation in ($\mathbf{x}$, $\mathbf{z}$)-vectors over multiple runs.

We summarize by stating that from different runs each initiated with a different random initial solution that there is a considerable variation in results from different runs. Moreover, the function evaluations needed to complete all grind sizes seem to be rather large.

4.2 Results Using Proposed Evolutionary Approach

We now present results from our proposed evolutionary approach which handles both the complexities mentioned above in a procedural manner. Thus, there is no need for multiple simulations with each grind size.

First, we present the results of 11 runs obtained with a population of size 40. The best, median, and worst objective function values in ounces and the same for number of function evaluations are presented in the first column of Table 2. It can be seen the results vary with 0.18% variation

Table 2: Variation of objective function (f) value and number of function evaluations over multiple runs.

	Original RGA		Heuristics RGA
Pop. size	40	200	40
Best (Oz.)	962,938	963,118	963,076
Median (Oz.)	962,465	962,935	962,800
Worst (Oz.)	961,225	962,714	962,728
Variance of f over runs	0.18%	0.04%	0.04%
Avg. Func. Evals.	1,680	8,400	1,640

in objective function value over 11 runs. This is much better compared to fmincon results which

had 2.27% variation.

To investigate the effect of population size on the variation of 11 runs, next we use a population size of 200. The second column of Table 2 summarizes the best, median and worst objective function value and number of function evaluations. It is clear that with a larger population size, the proposed evolutionary algorithm is better able to reduce the variance over multiple runs, thereby providing more consistent performance. The difference in optimized objective function value is only 0.04%. However, this consistency in function value comes with a cost of an increased number of function evaluations (8,400 versus 1,680, on an average). Next, we propose a derived heuristics based approach, which achieves a more consistent performance with a smaller function evaluations by exploiting the hidden knowledge in optimized solutions.

5 Derived Heuristics to Improve Algorithmic Efficiency

In this section, we introduce the concept of deciphering derived heuristics in an optimization problem. In many practical problems, certain heuristics describing problem information are usually available. In such cases, it is advisable to use the information to modify the optimization procedure. However, in many other practical problems, heuristics may not be readily available. The gold processing plant operation problem described above is one such problems, in which the operating principles are too complex to lend any such simplistic problem information. In such problems, heuristics can be derived by performing a repeated number of optimization runs and gathering useful information from optimized solutions. The idea is described in Figure 2.

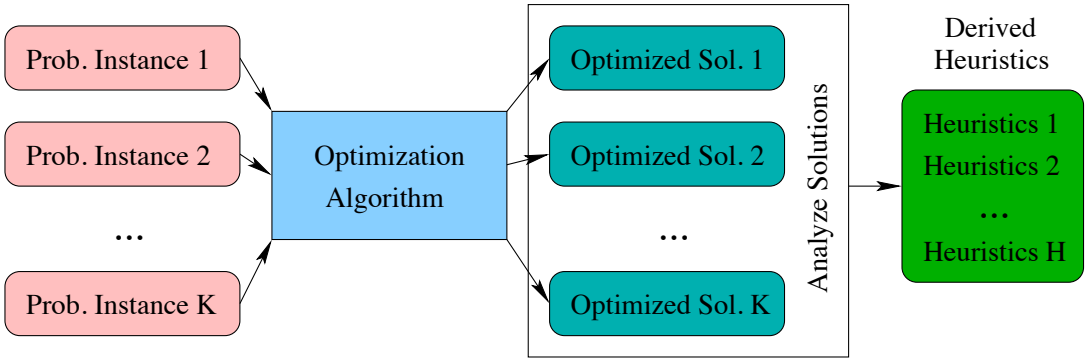

Figure 2: An illustration of the derived heuristics based optimization procedure.

The optimization problem is attempted to solve a few times (K times) for different input parameter setting of the problems. For this gold extraction process optimization problem, the input parameters are overall lithological compositions and grades of gold and sulphur contents in them. Once the optimized solutions are obtained for K such problem instances, they can be analyzed to decipher important derived heuristics (properties) of these solutions using manual means or by using any sophisticated machine learning procedures. Thereafter, a more reliable optimization method can be developed with these derived heuristics (H of them shown in the figure).

We have used a population size of 200 and run on seven lithology configurations ($K = 7$) shown in Table 3. It is interesting to note that all solutions ensure that either the capacity or power constraint of Mill 2 is exactly achieved. We use these hard constraints next. Some other invariant properties, we observe in the optimized solutions are as follows:

Table 3: Input parameter values for seven lithology configurations (L1 to L5).

Unit	Lith.	Prob. 1	Prob. 2	Prob. 3	Prob. 4	Prob. 5	Prob. 6	Prob. 7
Ton	L1	248,083	190,951	117,800	72,032	15,427	701,983	366,778
	L2	48,350	88,888	165,690	48,249	9,744	295,977	16,343
	L3	121,824	688,525	1,124,706	826,382	735,455	1,103,063	745,548
	L4	1,042,726	5,110,884	5,315,895	5,600,936	5,099,750	3,405,511	212,953
	L5	11,561,044	7,674,244	7,689,736	8,475,855	9,138,604	9,581,148	13,659,723
AU (g/t)	L1	2.68	3.45	2.74	2.96	2.61	2.34	1.87
	L2	2.91	3.94	3.31	4.14	4.76	2.41	1.40
	L3	3.84	3.48	3.59	3.76	3.75	3.45	3.67
	L4	3.30	2.54	2.50	2.44	2.44	2.44	2.05
	L5	2.60	2.59	2.68	2.49	2.66	2.44	2.22
S (%)	L1	5.90	4.39	5.51	4.03	4.18	7.58	6.39
	L2	6.15	6.41	6.14	4.98	6.25	7.50	6.63
	L3	3.32	4.48	4.42	4.52	5.11	5.78	5.00
	L4	5.25	5.97	5.93	5.98	6.00	6.02	5.65
	L5	3.80	3.29	2.98	2.89	2.99	2.59	2.76

- CIL process recovery, $\text{AU}_y = 0.91$ (Equation 8) for all optimized solutions. This happens due to Oxidation being set to its maximum $\text{OX}_{\max}$.
- The autoclave capacity constraint g_6 (Equation 6) is always active at all seven optimized solutions, meaning that the solutions take full advantage of the autoclave's available capacity.

These are interesting observations, as they provide better insights into the optimal operation of the plant and to know critical components of the overall process. Information such as optimized solutions utilizing maximum capacity of certain processes are valuable information for possible enhancement in productivity. Although the above information can stay as valuable heuristics of the problem, since they involve derived quantities and not the variables themselves directly, they will be difficult to use in repairing a solution.

When we analyze the optimized results we obtain valuable insights about the properties of the optimized solutions. A few interesting parameter properties are shown in Figures 3 to 4. The figures show that either Mill 2 power or Mill 2 capacity constraints are active at the optimal solutions of all seven problems, thereby meaning that Mill 2 is a crucial component in the specific gold processing plant. This information is not obvious prior to executing the optimization run. We obtain this information by first finding optimal solutions for each problem case and then analyzing the solutions.

Another vital information we gather from the results is that optimal variable x_5 (proportion of final lithology) takes a value within 0.4 and 0.7, as shown in Figure 5.

In our efficient evolutionary algorithm, we introduce these information in different ways. First, we set the lower and upper bound of variable x_5 within [0.4, 0.7] and for all other variables they are set within [0,1]. Second, we repair each random solution created in the initialization phase using information obtained for power and capacity constraints of Mill 2. In addition to fixing $\mathbf{x}$-vector for satisfying g_5 constraint in the initialization phase, we also check for capacity of Mill 2. If the tonnage capacity of Mill 2 has not reached by the $\mathbf{y}$-vector (obtained from $\mathbf{x}$-vector as follows: $\mathbf{y} = 1 - \mathbf{x}$), we then decrease $\mathbf{x}$-vector (thereby increasing $\mathbf{y}$-vector) to meet the Mill 2 tonnage capacity constraint exactly. In the event of a $\mathbf{y}$-vector exceeding the Mill 2 tonnage

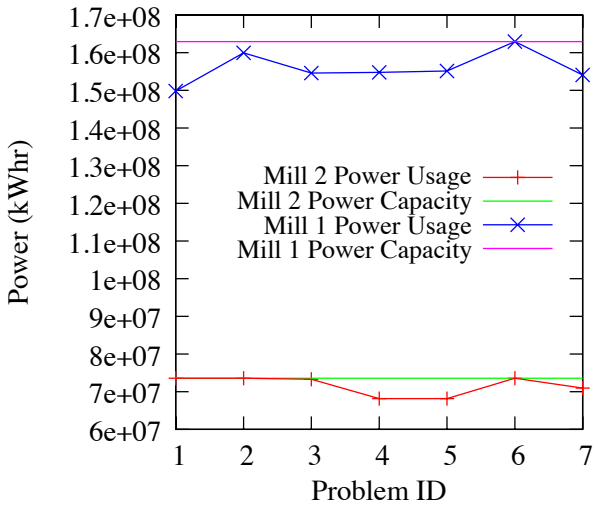

Figure 3: Power usage on both mills.

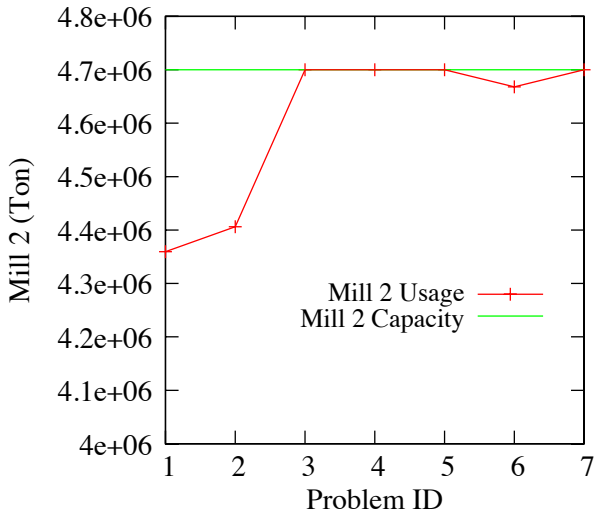

Figure 4: Tonnage capacity usage on Mill 2.

capacity constraint, we do not repair the solution in the initialization phase. This is because when this solution is evaluated subsequently, the excess material form Mill 2 gets diverted to Mill 1 by satisfying either of the Mill 2 power or tonnage capacity constraints. However, this change in initialization ensures that the either power or tonnage capacity of Mill 2 is always met.

5.1 Results with Derived Heuristics Based Approach

We implement the derived heuristics into the proposed evolutionary algorithm and use a population of size 40 to solve seven different problems. First, we present the results for the first problem, which is already solved using MATLAB's fmincon routine in Section 4.1.

First, the best, median and worst optimized objective function values for all 11 runs are shown in Table 2. It can be noticed from the table that these values, obtained with a population size of 40, are much better than those obtained using fmincon routine and is also comparable to the original evolutionary algorithm run with a large population size (of 200). Notice that the variation in optimized objective function values over 11 runs is only 0.04%, which is exactly same as that obtained by a large population size and without use of any heuristics. Thus, by using derived heuristics and a small population size, we are able to find similar quality solutions (with 0.04% variation over multiple runs) compared to a large population evolutionary algorithm but without any heuristics information, but using only a fraction (20%) of overall function evaluations (1,640 versus 8,400).

Figures 6 and 7 show the population-best and population-average objective function value as they vary with the generation counter for the first problem. The solid line is plotted with average of 11 runs and bars are shown for minimum and maximum values of 11 runs. It can be seen that both these values increase with generation and eventually stabilizes to the final optimized function value in all 11 runs.

For the same problem, we also show the optimal grind size in Figure 8 for one of the runs (terminating at generation 37) indicating that only two of the smallest six grind sizes (106 and 125 μ-m) are found to be optimal for this particular problem. To show the active constraints corresponding to the generation-wise best population members, we show all seven constraint values in Figure 9. The figure shows that two (g_4 and g_6) of the seven constraints are always

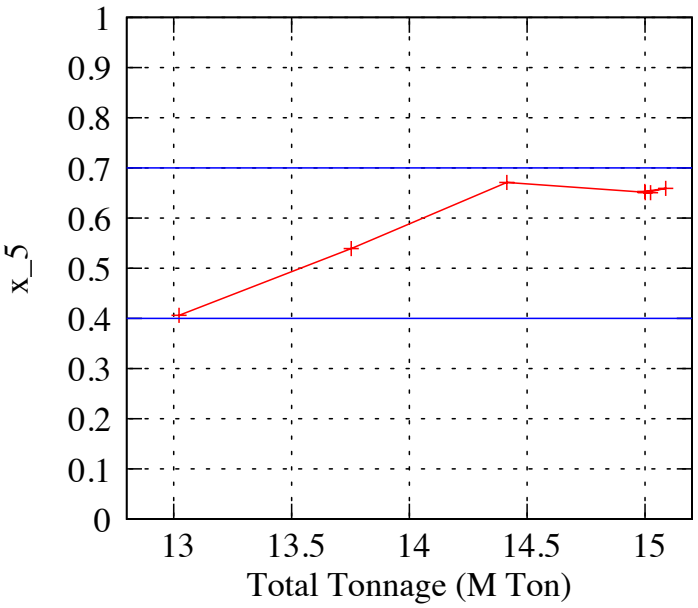

Figure 5: Variable x_5 for different problems having different tonnage of incoming materials.

active from start to finish of a particular simulation run. The constraint g_4 is the power capacity constraint of Mill 2 and by making this constraint active, the solution fully utilizes Mill 2's power capacity limit. On the other hand, by making constraint g_6 active, the solution ensures that the throughput mass is equal to the incoming amount of material to the autoclave. Although in intermediate generations the constraint g_3 (power capacity constraint of Mill 1) is active, at the final solution (at generation 37 of this particular run), Mill 1 power capacity is not maximally utilized, but other adjustments are made to make the solution better in terms of objective value. All other constraints are inactive at all generations of this particular run and not that important than constraints g_4 and g_6.

Table 4: Results of 11 runs of proposed heuristics based EA are shown.

Variable-vector **x**					Variable-vector **z**					Grind	Objective	Func.
x_1	x_2	x_3	x_4	x_5	z_1	z_2	z_3	z_4	z_5	(μ-m)	(Oz.)	Evals.
0.101	0.113	0.407	0.000	0.400	0.000	0.000	0.000	0.000	0.571	125	962,774.05	1,680
0.099	0.215	0.018	0.011	0.403	0.000	0.000	0.000	0.000	0.574	125	962,917.20	1,480
0.004	0.995	0.085	0.006	0.402	0.000	0.000	0.000	0.000	0.574	125	962,788.41	1,560
0.053	0.839	0.066	0.000	0.402	0.000	0.000	0.000	0.000	0.575	125	962,815.91	1,400
0.035	0.528	0.000	0.019	0.402	0.000	0.000	0.000	0.000	0.574	125	962,870.85	1,880
0.203	0.011	0.245	0.001	0.400	0.000	0.000	0.000	0.000	0.572	125	962,764.23	1,760
0.000	0.598	0.419	0.000	0.400	0.000	0.000	0.000	0.000	0.572	125	962,757.96	1,680
0.068	0.049	0.002	0.002	0.400	0.000	0.000	0.000	0.000	0.612	106	962,873.27	1,760
0.109	0.290	0.168	0.008	0.401	0.000	0.000	0.000	0.000	0.573	125	962,800.48	1,480
0.053	0.983	0.173	0.000	0.400	0.000	0.000	0.000	0.000	0.573	125	962,727.79	1,800
0.007	0.129	0.000	0.003	0.406	0.000	0.000	0.000	0.000	0.576	125	963,076.01	1,560

By comparing these results with those found using fmincon routine in Table 1, we notice that

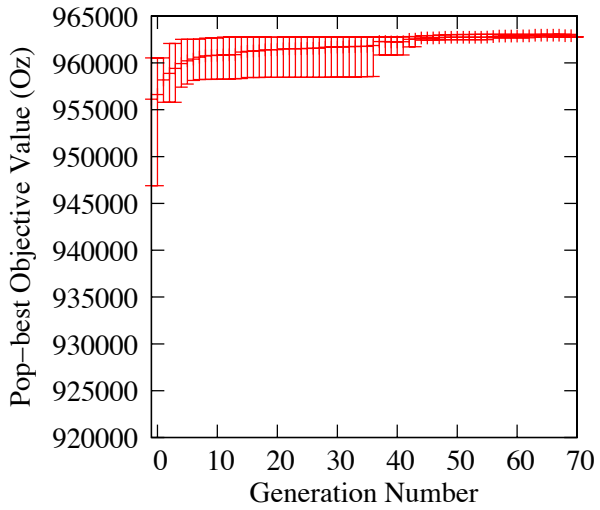

Figure 6: Population-best objective function value as it varies with generation. Variation over 11 runs is shown with error bars.

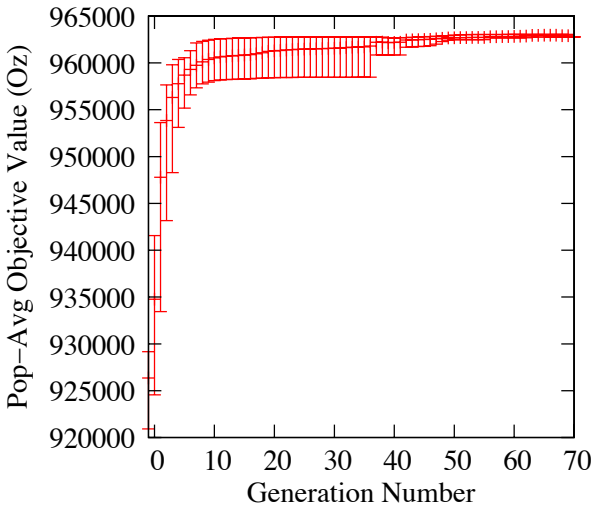

Figure 7: Population-average objective function value as it varies with generation. Variation over 11 runs is shown with error bars.

these results are more consistent from one run to the other. The excess amount from Mill 2 to Mill 1 (the **z**-vector) is very consistently zero for the first four lithologies and also the **x**-vector is more consistent between the runs. Since the fifth lithology (W_5) makes 89% of the overall input, the obtained solutions have a very similar x_5 value, thereby making the solutions quite consistent over multiple runs.

Now, we apply our heuristics based EA method to seven different problems for which the lithological configurations were presented in Table 3 and present results in Table 5. It is clear

Table 5: Performance of proposed heuristics based EA on seven problems having different lithological compositions.

Prob.	Best (Oz.)	Median (Oz.)	Worst (Oz.)	% Variation	% Varn. Original
1	963,016.52	962,862.46	962,749.98	0.028	0.536
2	1,011,533.88	1,010,474.99	1,010,043.11	0.148	0.271
3	1,068,779.76	1,067,166.54	1,066,486.58	0.215	0.489
4	1,040,434.15	1,039,024.79	1,038,287.12	0.207	0.680
5	1,075,617.55	1,074,965.97	1,074,271.31	0.125	1.169
6	1,033,431.52	1,032,743.14	1,030,556.08	0.278	1.470
7	897,403.38	896,981.34	896,704.88	0.078	0.140

from the table that in all seven problems, percentage variation in optimized objective function value is within 0.028% to 0.278%, which are much better than original RGA, shown in the last column of the table. The use of derived heuristics has helped to improve the performance of RGA. With such consistent performance of our proposed heuristics based EA, there is no need to run the method for more than one time, particularly when such a method needs to be called many times in a hierarchical and more global set up.

To validate how our proposed customized algorithm works on new problems, we create 10 new instances of input parameters randomly within the minimum and maximum values of material

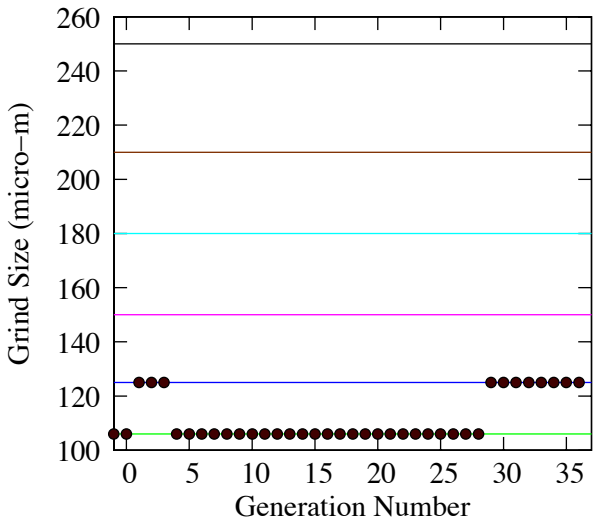

Figure 8: Grind size for the population-best solution as it varies with generation.

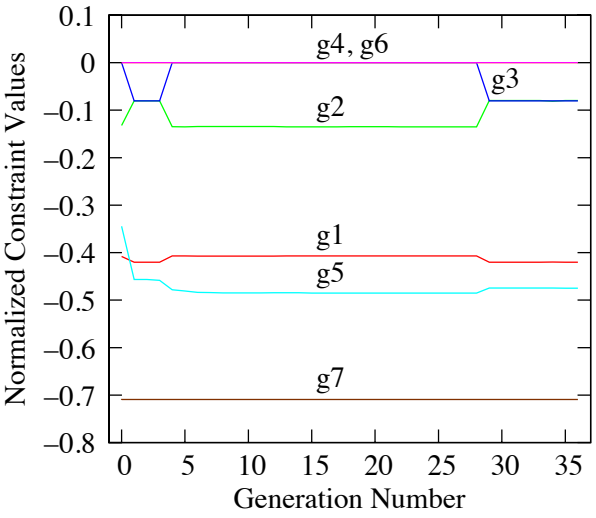

Figure 9: Constraint values of population-best solution with generation.

composition used in the above seven cases and apply our method. Table 6 shows that compared to original RGA approach, our heuristics-based customized RGA approach produces a smaller variation in objective values over 11 runs in each of 10 cases. The median values of objective

Table 6: Validation of proposed heuristics based RGA on 10 new instances. Compared to original RGA, heuristics-based RGA produces more consistent results over 11 runs. Results with a smaller variation in f values over 11 runs are marked in bold.

		Problem									
		1	2	3	4	5	6	7	8	9	10
Orig.	(Oz.)	930,886	968,913	687,842	1,081,647	965,050	1,088,621	937,366	982,436	931,998	937,983
	(% diff)	0.52%	0.38%	0.41%	0.64%	0.42%	0.39%	0.30%	0.06%	0.46%	0.29%
Heur.	(Oz.)	932,059	968,776	688,412	1,081,030	965,375	1,088,659	936,700	982,508	931,576	938,095
	(% diff)	**0.37%**	**0.31%**	**0.29%**	**0.26%**	**0.13%**	**0.21%**	**0.19%**	0.06%	**0.34%**	**0.22%**

values for the heuristics-based RGA approach is equivalent to those for the original RGA, but is found to be better than those for the original RGA for most of the problem cases.

6 Conclusions

Evolutionary optimization methods, such as genetic algorithms, differential evolution, evolution strategies, and other nature-inspired optimization algorithms are getting increasingly popular in practical problems due to a number of unique properties and flexibilities that they provide. In this paper, we have attempted to solve a real-world gold processing plant operation problem involving nonlinear and non-differentiable operating principles. We have provided a detail formulation of the resulting optimization problem.

First, we have applied MATLAB's fmincon routine – a popularly used optimization methodology – in an attempt to solve the problem. Certain practicalities associated with the resulting problem have forced us to solve the problem sequentially multiple times to obtain the optimized solutions. Over 11 different simulations starting from different initial solutions, MATLAB's fmincon routine has found a 2.27% variation in the optimized objective value. Although, this is not particularly a large variation, when such a process optimization is embedded within a higher level design process and when the optimization needs to be performed many times in a nested manner, a more consistent achievement of the optimized solution with a single run is expected. In addition to the variation, it has also been observed that the optimized solution vectors obtained in different runs are significantly different from each other.

Second, we have implemented the optimization problem formulation in a real-parameter genetic algorithm (RGA) code and solved. Results from 11 different runs with a small population size have indicated a reduced variation of optimized objective function value over runs to only 0.18% (compared to 2.27%). The same RGA with a large population size has managed to reduce the variation of optimized objective value over multiple runs to a mere 0.04%, thereby exhibiting consistent performance of the algorithm. But due to the use of a large population size, the computational effort is large.

Third, we have demonstrated the use of derived heuristics based optimization method proposed in this paper to improve the performance of the RGA with a small population size. For this purpose, first, we have studied the optimized results from the small population sized RGA runs and gathered heuristics hidden in the optimized solutions. A few such heuristics have then been embedded in RGA's initialization phase and also in re-setting the bounds of a critical variable of the problem. The problem is then re-solved 11 times with the derived heuristics based approach. It has been observed that by using the derived heuristics, RGA has been able to find same quality and consistent solutions as the large population original RGA runs did, but using only a fraction (20%) of computational effort. This clearly demonstrates the usefulness for customizing a generic optimization algorithm for solving a practical problem for a better and a more consistent performance, despite the lack of any obvious heuristics directly before the optimization task is performed.

Fourth, the heuristics based RGA has been finally applied to seven different problems and 10 different new problem instances and a better consistent performance of the algorithm has been reported for each instance.

This paper has clearly demonstrated the usefulness of using derived problem information in improving the performance of an optimization algorithm. In problems where problem information is not readily available, they can be achieved by analyzing optimized solutions for a number of input instances of the problem. The paper has shown the discovery and use of derived heuristics in a generic optimization algorithm for improving the performance of the algorithm through a real-world optimization problem. Since evolutionary optimization algorithms allow an easy embedding of any kind of heuristics, this paper has used such a method on a typical real-world problem solving task. For such routine applications, a heuristics based optimization method demonstrated here is expected to produce more consistent results than the original stand-alone methods. The heuristics based methodology described in this paper is generic and can be applied to other more challenging real-world problems.

Acknowledgment

Authors acknowledge the support provided by Maptek Private Limited, Northbridge, Western Australia for carrying out this study.

References

[1] K. A. Al-shayji, S. Al-wadyei, and A. Elkamel. Modelling and optimization of a multistage flash desalination process. *Engineering Optimization*, 37(6):591–607, 2005.

[2] A. Aravelli and S. S. Rao. Energy optimization in chiller plants: A novel formulation and solution using a hybrid optimization technique. *Engineering Optimization*, 45(10):1187–1203, 2013.

[3] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: Theory and algorithms*. Singapore: Wiley, 2004.

[4] D. W. Corne and J. D. Knowles. No free lunch and free leftovers theorems for multiobjective optimisation problems. In *Proceedings of the 2nd international conference on Evolutionary multi-criterion optimization*, EMO'03, pages 327–341, Berlin, Heidelberg, 2003. Springer-Verlag.

[5] K. Deb. *Optimization for Engineering Design: Algorithms and Examples*. New Delhi: Prentice-Hall, 1995.

[6] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2–4):311–338, 2000.

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

[8] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[9] K. Deb and C. Myburgh. Breaking the billion variable barrier in real-world optimization using a customized evolutionary algorithm. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2016)*, pages 653–660. ACM Press, 2016. DOI:http://dx.doi.org/10.1145/2908812.2908952.

[10] W. R. Ellingsen. Operating cost optimization of a batch process. *Engineering Optimization*, 10(2):85–95, 1986.

[11] D. E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[12] J.-S. Kang, M.-H. Suh, and T.-Y. Lee. Robust economic optimization of process design under uncertainty. *Engineering Optimization*, 36(1):51–75, 2004.

[13] C. Li, X. Wang, and L. Tang. Operation optimization in the hot-rolling production process. *Ind. Eng. Chem. Res.*, 53(28):11393–11410, 2014.

[14] A. Mari, C. Mattson, A. Ismail-Yahaya, and A. Messac. Linear physical programming for production planning optimization. *Engineering Optimization*, 35(1):19–37, 2003.

[15] S. T. A. Pratomo. Optimization gold cyanidation process to increase gold extraction at pongkor gold mining business unit indonesia. *Mine Planning and Equipment Selection*, pages 1081–1089, 2014.

[16] S. S. Rao. Genetic algorithmic approach for multiobjective optimization of structures. In *Proceedings of the ASME Annual Winter Meeting on Structures and Controls Optimization*, volume 38, pages 29–38, 1993.

[17] G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell. *Engineering Optimization Methods and Applications*. New York : Wiley, 1983.

[18] Y. Saif, A. Elkamel, and M. Pritzker. Superstructure optimization for the synthesis of chemical process flowsheets: Application to optimal hybrid membrane systems. *Engineering Optimization*, 41(4):327–350, 2009.

[19] A. Solonen and H. Haario. Model-based process optimization in the presence of parameter uncertainty. *Engineering Optimization*, 44(7):875–894, 2012.

[20] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.