Feasibility and Cost Minimisation for a Lithium Extraction Problem

P. Bosch[a], J.P. Contreras[a], J. Saavedra-Rosas[b,c,*]

[a]Facultad de Ingeniera, Universidad del Desarrollo, Santiago, Chile [b]Universidad de Chile, Facultad de Ciencias F´ısicas y Matemáticas, Departamento de Ingenier´ıa de Minas, Santiago, Chile [c]Curtin University, Department of Mineral and Energy Economics, Perth, WA, Australia

Abstract

In this paper we address the problem of allocating extraction pumps to wells, when exploiting lithium rich brines, as part of the production of lithium salts. The problem of choosing the location of extraction wells is defined using a transportation network structure. Based on the transportation network, the lithium rich brines are pumped out from each well and then mixed into evaporation pools. The quality of the blend will be based on the chemical concentrations of the different brines, originating from different wells. The objective of the problem is then to determine a pumping plan such that the final products have predefined concentrations, and the process is operated in the cheapest possible way. The problem is modelled as a combinatorial optimisation problem and a potential solution to it is sought using a genetic algorithm. The evaluation function of the genetic algorithm needs a method to determine feasible minimum cost flows for the proposed pumping alloca- tion, thus requiring the formulation of a blending model in a flow network for which a new iterative non-convex local optimisation algorithm is proposed. The model was implemented and tested to measure the algorithm's efficiency.

Keywords: Optimisation, Feasibility, Mine Planning, Lithium, Non-convex Optimisation

*Corresponding author
Email addresses: pbosch@udd.cl (P. Bosch), juan.contrerasff@usach.cl (J.P. Contreras), jsaavedra@ing.uchile.cl (J. Saavedra-Rosas)

Preprint submitted to Computers & Operations Research February 26, 2019

1. Introduction and motivation

New mobile technologies such as digital cameras, notebooks and mobile phones are essential components of modern life. However, regardless of which equipment is being used, its operational capability is limited by the quality of the batteries used to power it. Increasing battery life has motivated research of new technologies to store energy. Among several new options for energy storage, fabrication of lithium based batteries has become popular, this has been mainly motivated by the properties of this element. Lithium is one of the lightest elements of the periodic table and it is capable of providing a high electric potential, properties that have transformed it into a highly consumed and demanded product. A good source of lithium can be found in salt flats. Some of the most important deposits in the world are located in Bolivia (Uyuni), northern Argentina (Hombre Muerto), Israel (Dead Sea), United States (Great Salt Lake, Silver Peak, Searle Lake and northern Chile (Salar de Atacama). The Atacama salt flats are the biggest in Chile with an approximate extension of 300 square kilometres, it is located in a valley between the Andes Domeyko moutain ranges. This particular salt flat is composed by big quantities of gypsum and salt rocks. The salt rocks are continuously fed by brine with a 28-47 parts per million (ppm) concentration coming from the Salado and San Pedro rivers [16]. The extraction process consists in pumping out brine from the salt flat using shallow surface wells, it needs to be noted that pumping out brine from a well requires the use of a pump that needs to be placed on the well. The extracted brine, when available from the well, is saturated in salt and gypsum with high concentrations of Na^+, K^+, Mg^{+2}, Li^+, Ca^{++}, SO_4^{-2} y Cl^- among others [15]. In the case of Salar de Atacama, there are more than 200 wells enabled and around 90 available pumps that can be operated simultaneously to per- form the extraction process. The chemical characteristics of each well are not constant and change according to different properties such as depth or porosity of the soil, just to mention a couple of them. The constant input of rivers, and the same extraction process, produce changes in the chemi- cal properties of the wells, which makes regular measurement of the those properties essential for the operation of the extraction method. Finally, the extracted brine is sent (by means of pumping) into evaporation pools where different processes such as evaporation or decantation are used to obtain the

final products following specific chemical specifications. Given the disparity in the nature of the wells, chemical properties and pump capacities, it is possible that the mixture that is created in the evapora- tion pools (also called terminals or sinks), fails to provide the desired chemical properties and concentrations in the final products. To avoid the occurrence of this problem, intermediate accumulation pools that sit between the ex- traction wells (sources) and the evaporation pools (sinks) are used. These intermediate pools enable mixtures that increase the chances to obtain the required concentration in the sinks. The pumping of brine requires the use of energy which translates into costs that the companies using this extraction technique have to pay. Due to different characteristics, different extraction wells will require different energy quantities used to transport the brines. It is desirable for the company to obtain a final product, within specified specifications, with minimum production cost. Figure 1 shows a schematic representation of a typical operation. It can be observed that the different elements such as extraction wells, connect- ing tubes, accumulation and evaporation pools conform a network of inter- operating elements that allow the flow of brines from the salt flat to the final destination where the product is produced.

Figure 1: Representative diagram of the network flow (sectional cut)

The general problem considered in this paper is to determine the set of wells in which extraction pumps are going to be located, to create an extraction network together with an extraction schedule. This should be done in such a way as to obtain a flow satisfying chemical requirements in the final product and ideally at a minimum cost of production. The problem thus formulated can be decomposed into two main elements: feasibility and optimality. The first component, feasibility tries to obtain an extraction schedule that is able to produce final product with the desired

characteristics. The second problem looks at the cost component of the operation of the system. For the purposes of this study, the problem has been decomposed similarly into two components. One component uses a non-convex optimisation algorithm to determine feasible flows when the location of the pumps has been determined. The feasibility component is then called by an optimisation procedure, that tries to obtain the cheapest possible way to operate a feasible flow, based on the current characteristics of the wells

and available pumps.

Each individual of population is a fixed network

The fitness function is the minimum cost flow on each fixed network

Minimise Cost

Feasibility problem

Minimise Chemical feasibility Error

Feasible flows of the network

Figure 2: Representative diagram of the structure of the algorithm

The remainder of this paper is organised as follows: In section 2 we per- form a literature review and analyse classical pooling problem formulations over a fixed network. In section 3 we develop a new model that considers specific requirements present in extraction of Lithium rich brines(represented in figure 2 as the Feasibility Problem box), and we establish an algorithm for local optimisation for a given arrangement of extraction pumps, where the total cost of the operation is proportional to the amount of brine moved trough the network. This optimisation algorithm uses the feasibility problem and approximates the final concentrations adding cost constraints (represented in figure 2 as the Flow in fixed network box). In section 4 the network topology

problem is considered and approached using genetic algorithm (GA) utilising the feasible flow algorithm defined before. The GA calls the algorithm presented on section 3 to assess the feasibility of a proposed arrangement of pumps being evaluated (see figure 2). In section 5 numerical tests run over a simulated instance with 90 extraction wells, 8 mixing pools, 6 evaporation pools and 10 components are presented. Finally, in section 6 we conclude and present some possible extensions.

2. Related literature

Blending problems with cost minimization have been largely studied under the distinctive name of pooling problems. In [18] pooling problems are described as a mix between blending problem and classical network flow problems. Three types of resources are distinguished in the network: source containing material with a known chemical specification, intermediate pools used for accumulation and mixing, and sinks where material is blended into a specific quality specification. The usual objective in pooling problems is to determine a minimum cost plan to flow material within the network such that final blend specifications are satisfied. The pooling problem is very important in the petrochemical industry context. Nevertheless, its general formulation can be adapted to other application areas such as waste-water treatment, paint industry or emissions control. More details about application areas for this problem can be found in [21]. In this paper, a novel application of pooling models has been proposed for Lithium industry. The first mathematical nonlinear formulations were introduced by [19], for this model which uses specification variables, corresponds to the most intuitive model and its know as p−formulation. Later, newer modelling options were proposed, for example the q−formulation was proposed in [7] and [27] replaced the specification variables by proportion variables which denote the fraction of incoming flow from sources to mixing pools. The pq−formulation proposed in [32], incorporates some extra and valid inequalities derived from a reformulation-linearisation technique into the q−formulation. Also, a hy- brid formulation that combines specification and proportion variables can be find in [4], where the proposed model extends the q−formulation. The same author defines generalized pooling problems where connections between pools are permitted. In [23], the model became more general and included the topology of the decision network. Pooling problems are known to be NP-hard and all the models above are equivalent, a complete survey about

different models can be found in [17]. Some points are common for all formu- lations: classical flow constraint are used to model material transport trough the network, objective function is linear and represents the cost of transport- ing material through the network, or can represent profit associated with the sale of products obtained in terminal sinks. Upper bounds are used to limit incoming flow into the network resources. Bilinear constraint are required to describe chemical specifications in pools and final blends, those last ones being also involved in range constraints.

Lithium applications requires some modifications with respect to the clas- sical formulations of the pooling problem. In particular, in this paper we consider demand constraints in final blends. Demand constraints force po- tential solutions to the problem to bring flow in all the terminal sinks, and at the same time all the chemical specification constraints in the problem must be satisfied. This represent a departure with respect to the more classical pooling problem formulations, because in the standard pooling problem a flow equals to zero is always a feasible solution for which specification con- straint are trivially satisfied. As mentioned in [29], using demand constraints to find a feasible solution makes the problem harder, however, the feasibility domain for the problem gets smaller and it might be easier find an optimal solution using exact methods.

Several approaches to solve pooling problems have been proposed using local and global optimization techniques. Some local optimization techniques include successive linear programing (SLP) [31, 5], here bilinear constraints are linearised using Taylor's expansion and a sequence of strategic linear programs (LPs) are solved. In [4], a branch-and-cut quadratic algorithm is proposed, also new variable neighborhood search heuristics (VNS) are de- veloped, and then a comparison of this method with the SLP method is provided. Methods that approximate bilinear constraints, such as the one found in [26] are also found in the literature, in this work the author discre- tises quality variables, whilst in [2] the discretisation is done in the domain of proportion variables. Global optimization efforts include: generalized Ben- der's descomposition [12] and Lagrangian-based methods [3, 1]. Applications of general methods like global optimization algorithm (GOP) defined in [33], approximate a global solution through a series of primal and relaxed dual problems. Also, different branch-and-bound or branch-and-cut procedures have been proposed, see for example [27], where a relaxed LP is proposed and used in a spatial search. In [13], convex approximations of the bilinear terms are investigated. A more detailed and complete survey about tech-

niques to solve pooling problems can be found in [18].

3. Flow in a fixed network

The transport network is modelled as a directed graph $G = (V, A)$, defined by a set of nodes $V = S \cup I \cup P$, where S, I, P are disjoint sets which correspond to extraction wells, accumulation pools and evaporation terminals respectively. In the set A of edges for the graph, the only pairs that are found are those that connect nodes of S with nodes of I, and those that connect nodes in I with nodes in P, no direct arcs between sources and terminals are permitted.

$$A = \{(s, i) : s \in S, i \in I\} \cup \{(i, p) : i \in I, p \in P\} \quad (1)$$

For each accumulation pool it is considered that there is a minimum incoming flow ($\varepsilon > 0$), otherwise the existence of the pool would not be justified. The variable $f_{u,v}$ denotes the flow being moved from node u to node v. The condition $f_{u,v} \geq 0 \ \forall (u, v) \in A$ indicates that the flow is unidirectional. The following constraints are introduced into the model:

• (C1) Flow conservation: $\sum_{s\in S} f_{s,i} - \sum_{p\in P} f_{i,p} = 0 \quad \forall i \in I$

• (C2) Available capacity in sources: $\sum_{i\in I} f_{s,i} \leq F_s^{max} \quad \forall s \in S$

• (C3) Minimum flow required in terminals: $\sum_{i\in I} f_{i,p} \geq F_p^{min} \quad \forall p \in P$

• (C4) Minimum flow required in accumulation pools: $\sum f_{s,i} \geq \varepsilon \ \forall i \in I$ $_{s\in S}$ The set of feasible flows of the network is thus defined by the satisfaction 7
of

these four constraints and parametrised by ε: $\Phi_\varepsilon =$

□□□

$\sum f_{s,i} \leq F_s^{max} \ \forall s \in S \ \sum_{i\in I}$

$_{s\in S}$□□□□□□□□□□□□□□ $f_{s,i} - \sum_{p\in P} f_{i,p} = 0 \ \forall i \in I$ $f \in R^{|A|}$

$_+ : \sum_{i\in I}$
(2)
3.1. Feasibility flow The problem currently modelled in this first stage is a feasibility problem, i.e., our objective is to find a flow creating a mixture of chemical solutions in the evaporation nodes, where the expected concentrations are obtained in those nodes. Some mathematical transformations and operations are intro- duced in order to model the feasibility problem as a conditioned least squares problem, and then use classical non-linear optimization techniques to solve it. In what follows, E denotes the set of chemical products present in the mixture. On each node v ∈ V of the

network, a variable $z_{v,e}$ is defined which denotes the concentration of the component e present in that particular node. The initial concentrations in the source nodes can be measured and they will be considered being data for the problem and denoted by

$\hat{z}_{s,e}$. A natural condition is then imposed: $z_{s,e} = \hat{z}_{s,e} \ \forall \ s \in S, e \in E$ (3)
The concentration of components in pools and terminals can be deter- mined uniquely from the flow and initial concentrations by means of a mass

balance (in absence of chemical reactions of the components) $z_{i,e} =$ $f_{i,p} \geq F_p^{min} \ \forall p \in P$

$\sum_{s\in S} f_{s,i} \geq \varepsilon \ \forall i \in I$

$\sum z_{s,e} f_{s,i}$ $\sum_{s\in S} \sum_{i\in I}$

$_{s\in S} f_{s,i}$

$z_{i,e} f_{i,p} \ \forall i \in I, e \in E \wedge z_{p,e} =$
$\sum \forall p \in P, e \in E$ (4)

$_{i\in I}$

Defining $Z = (z_{v,e})$ as the matrix that contains all the concentration va- riables, then the initial condition (3) and the equations (4) can be written 8

$f_{i,p}$

more

concisely (in matrix form) as: $L(f)Z =$

]

(5)

where L is an operator that associates to each flow a square matrix (lower triangular) whose elements are $l_{n,m}(f) =$

[$\overset{□}{Z}_S$

$0_{(|V|-|S|)\times|E|}$

□□□□□□□□□□□□

1 $\sum$ if $m = n, n \leq |S|$ $_{u\in V}$ $f_{u,n}$ if $m = n, n > |S|$

$-f_{m,n}$ if $m<n$ 0 otherwsise

(6)

Being L a lower triangular matrix, its determinant can easily be computed as the product of the elements on its diagonal. Using also constraints (C3) and (C4) we obtain the following expression for the determinant:

$\det(L(f)) = \prod$

$_{v\in V-S}$

$(\sum f_{u,v})$

$\geq \varepsilon^{|I|} \prod_{u\in V}$ $_{p\in P}$

$F_P^{min} > 0$

hence, the operator L is invertible ($\det(L(f)) = 0$) and the concentration variables can be expressed uniquely in terms of flows and initial concentra-

tions $Z(f) = L(f)^{-1}$ [$\hat{Z}_S$

$0_{(|V|-|S|)\times|E|}$

]. (7)

On each terminal it is expected that a final product with a pre-specified chemical composition can be obtained. If we denote by $\hat{z}_{p,e}$ the concentration of component e expected in terminal p, we are then interested in those flows f such that $z_{p,e}(f)=\hat{z}_{p,e}$ $\forall$

$p \in P \; e \in E$ (8)

The previous condition can be expressed in matrix form as $Q_P Z(f) = \hat{Z}_P(f)$ (9)

where $Q_P = [0_{|P|\times(|V|-|P|)} \; Id_{|P|}]$, then the concentration variables in the terminal $Z_P(f)$

nodes $:=$ whilst $Q_P Z(f)$ $\hat{Z}_P(f)$ corresponds is the matrix to

$|P|\times|E|$ that groups the elements $\hat{z}_{p,e}$. 9

It is proposed that the following non-linear optimisation problem is solved
to find flows satisfying the condition expressed by equation (9) min $H(f) :=$

$\| \| \| Z_P(f) - \hat{Z}_P$ s.t.

$\| \| \|^2_F \; f \in \Phi_\varepsilon$

(10)

here $\cdot_F$ represents the Frobenius matrix norm, with the flows of inte- rest being
those such that $H(f) = 0$. The objective function, being non convex, could result in
local solutions to the optimisation problem for which $H(f) = 0$, in these cases only an
approximation to the desired concentrations is obtained. The function $H(f)$ is

differentiable for all $f \in \Phi_\varepsilon$ and its partial deriva-
tives are given by the formula: $\partial H(f)$

$\partial f_{u,v} = \mathrm{tr}\left(\left(\hat{Z}_P - Z_P(f)\right)\left(Q_P L(f)^{-1} \partial L(f)\right.\right.$

$\partial f_{u,v}$))

$Z(f)$(11)

where tr(.) represents the trace of a matrix derivatives of the components of $L(f)$, more

and precisely $\partial L(f)$

$\partial f_{u,v}$ is the matrix of the $\partial L(f)$ $\partial f_{u,v} =$

$(\partial l_{m,n}$

$) f_{u,v}$

$_{N\times N} \wedge \; \partial l_{m,n}$

$f_{u,v} =$

1 , is $n = v, m = v - 1$,if $m = u$, $n = v$

0 , otherwise

(12)

The calculation of the gradient of the objective function allows the use

of which $\hat{f}^{m}$ classical is obtained is a method non-linear as the of directions solution optimisation of f^{m+1} the techniques following $= f^{m} + \alpha$such $_{m}$(linear $\hat{f}^{m}$ as $-f$problem: Frank-Wolfe $_{m}$) where the method, vector min $\nabla H(f^{m})\, f$

s.t.

$f \in \Phi_{\varepsilon}$

(13)

On each iteration, the size of the step α_m can be chosen using an Armijo rule. Of course, different direction methods and step size rules can be used to solve the problem, see for example [8] and [6]. 10

3.2. Incorporating Cost The movement of flows through the network requires an important expenditure of energy, which directly translates into economic costs for the company exploiting the salt flat. This cost is a variable one because it depends on the flow being moved. We must point out that obtaining a flow that satisfies the demand constraint and chemical specifications - in evaporation nodes - is important but not enough, because a solution having an excessive cost to it, is not deemed practical alternative. It has been natural to model the cost function components for the problem as linear ones [17]. Under this modelling paradigm, the total cost of the operation will be proportional to the amount of brine moved trough each element of the network. There are elements that are costlier than others (depending on distances, altitude with respect to the sea level, etc.). Let us denote $c_{u,v} > 0$ as the cost coefficients that indicate the cost of moving one flow unit using the arc $(u, v) \in A$ in the network, hence the total cost is given and noted as

$$C f = \sum_{(u,v)\in A} c_{u,v} f_{u,v} \quad (14)$$

In an ideal situation, the problem that we would like to solve is: min $C f$ s.t.

$$f \in \Phi_{\varepsilon}\ H(f)=0$$

(15)

which is simply cost minimisation subject to flow feasibility constraints. However, constraint $H(f) = 0$ is a difficult one to achieve due to the non-convex nature of the

function H. To search for solutions that approximate product requirements and have a minimal cost, we propose a method that exploits the linearity of the objective function and use the idea developed in the previous section to obtain feasible flows. The proposed method is iterative and works in the following way: 1. On iteration $k = 0$ a minimum cost flow is obtained $f^{(0)}$ that solves
the following linear problem LP min $C f$
s.t.
$f \in \Phi_\varepsilon$
, (16)

let σ^* denotes the value of the minimum cost $C f^{(0)}$. 2. For iteration k, the flow $f^{(k-1)}$ of the previous iteration is used as a
starting point for the Frank-Wolfe algorithm to solve the problem min $H(f)$
s.t.

$$f \in \Phi_\varepsilon \; C f \leq (1 + \alpha_k)\sigma^*$$

(17)

3. If $C f^{(k)} < \sigma^*(1 + \alpha_k)$ or $H(f^{(k)})$ is small enough, then the method finishes providing $f^{(k)}$ as a solution. Otherwise, we return to point 2 for iteration $k + 1$. The sequence of positive parameters α_k is chosen to be increasing, in a way such that $\lim_{k\to\infty} \alpha^k = +\infty$, however the growth rate for the parameter should decrease from one step to the other. One possible option is to build

the parameters as $\alpha_k =$

$$\sum_{j=1}^{k}$$

a_j (18)

where $(a_j)_{j\in N}$ is a sequence converging to zero but whose series diverge, for

example $a_j = 1/j$. The intuitive idea of the method is to approximate the final concentrations on sets for which the cost is bounded. On each iteration the cost increases allowing obtaining a better approximation of the required concentrations on the final product. Also, the growth of the cost bound is smaller on each step allowing for a finer search. The method stops when an acceptable approx- imation is obtained, this is when $H(f^{(k)})$ is small, or when the cost bound is not active in problem given by equation (17). In this last case, we are in presence of a local minimum for the problem and there are no directions for which the search process could continue. The previous statement and some properties are justified in the following theorem. Theorem 1. Let $\{f^{(k)}\}$ the sequence generated by the iterative method, then i. If $f^{(k)}$ does not activates the cost constraint $C f \leq (1 + \alpha_k)\sigma^*$, then it is a local minimum of H over whole space Φ_ε. ii. The iterative algorithm finishes. Also, if k is the first value for which $H(f^{(k)}) \leq H_{tol}$, then the cost of $f^{(k)}$ is at most $(\alpha^k - \alpha^{k-1})\sigma^*$ units 12

to

$H(f) \leq H$

$\in \Phi_\varepsilon$

bigger

(19
)

f a local optima for the

ise C f

subject

Proof. i. This part is clear since φ_ε is convex and constraint $C f \leq (1 + \alpha^k)\sigma^*$ is a cut. If $f^{(k)}$ is a local minimum of problem (17) and the constraint is not active, then no feasible descend directions of H over φ_ε can be found, and therefore is a local minimum of H over whole space Φ_ε. ii. For the second item, we know Φ_ε is compact due to the capacity con- straints in the wells, then $\max\{C f : f \in \Phi_\varepsilon\}$ exists. As $\alpha^k \to \infty$, at some point the cost constraint is irrelevant and it wont be activate, which is one of our stopping criteria. Finally, if k is the first non-negative integer for which $H(f^{(k)}) \leq H_{tol}$ we have $C f^{(k-1)} = (1+\alpha_{k-1})\sigma^*$ because the algorithm does not stop in $k - 1$, and $C f^{(k-1)} < C f^{(k)}$ because $f^{(k)}$ is not attainable at iteration $k - 1$. Denote by f^* a local optimum of (19), then clearly $H(f^*) \leq H_{tol} < H(f^{(k-1)})$, and $C f^{(k-1)} < C f^* \leq C f^{(k)}$ (20)

because f^* is not attainable at iteration $k-1$. Join the results we have

$$(1 + \alpha_{k-1})\sigma^* \leq C\, f^* \leq C\, f^{(k)} \leq (1 + \alpha_k)\sigma^*$$

from where it is easily obtained
that

$$C\, f^{(k)} \leq C\, f^{\,*} + (\alpha_k - \alpha_{k-1})\sigma^*$$

D

4. Choosing the Network: Genetic Algorithms

The problem of choosing the extraction wells consists in determining which wells (out of all the possible set of wells) will be selected to build the definitive network flow. Given that there are more wells than pumps

available to operate simultaneously, the problem is of a combinatorial nature and we will use heuristic techniques to solve it. Between two different wells the main two differences are: extraction cost and chemical properties of the brine that can be extracted from them. In the previous section, a method was proposed to determine flows that provide final products satisfying chemical requirements at minimum cost. In this section, we will combine the method described previously with a genetic algorithm (GA) to evaluate different network flow configurations and approximate an optimal selection of the network configuration[1]. Let S be the set of all the available wells with $|S| = N$ and the whole network $G = (S \cup I \cup P, A)$. Let M be the quantity of extraction pumps that can be operated simultaneously, we want to determine a subset S of S such that $|S| = M$

and the network $G(S)=(S \cup I \cup P, A|_S)$, which is the sub-network using only the wells provided in S, be capable of providing a feasible flow at minimum cost. Each time a subset S from S is fixed, a sub-network is obtained for which a minimum cost flow can be sought that approximate the desired requirements for the final product using the iterative method presented in section 3.2. This mechanism provides an evaluation system for any choice of wells and potentially allows the use of other heuristic optimisation methods. Genetic Algorithms, originally proposed by J. Holland [20], are methods that are able adapt to different problems in search and optimisation. They are inspired in the Darwinian evolutionary process for live organisms, in particular, natural selection and survival of the fittest. GAs use the natural selection process as the key driver for an adaptive search of good solutions to a given problem. It starts with a selection of a representation of potential solutions to a problem (encoding) and from there an initial population is generated (where each individual is a potential solution to a given problem), those individuals are evaluated by means of a fitness function (or objective function) and submitted to a selection process that will define whose individuals will pair to produce descendants (crossover and mutation).

[1]It is important to mention here that GAs do not provide a certificate of optimality but they are generally used as an alternative in the context of difficult combinatorial problems, which motivates our choice.

4.1. Proposed Encoding Encoding is a fundamental block in GAs. Each possible solution to the problem needs to be encoded as an array of genes (data) and, ideally, each chain of genes should correspond to a possible solution. For the wells selection problem the feasible solutions are subsets of S with M elements, so we need an encoding that represents such subsets. Lam [22], proposed an encoding with pigeon-hole coding scheme for solving sequencing problems which is suitable for being applied in our context of pump allocation.

Let S = N). To represent $\{s_{i1}, ..., $ the $s_{iM}\}$ subset a subset of of S selected = $\{s_1, ..., s_N\}$

with M elements (M wells S through the pigeon-hole $<$ encoding we use an array of M

entries. The array components $[p_1, ..., p_M]$

are chosen according to the following rule: $p_1 = i_1$

$p_k = i_k -$

$\sum_{j=1}^{k-1}$

$\varphi_k(i_j)$ $k > 1$ (21)

where φ_k is such that $\varphi_k(i_j)$ =

$\{$ 1, if $i_j < i_k$

0, otherwise (22)

To better illustrate this coding scheme, a toy example will be considered. Suppose we want to encode the selection S = $\{s_2, s_3, s_6, s_8\}$, i.e. the wells 2, 3, 6 and 8 are selected from a total of N = 9 possible allocations for pump installation. We start with a complete list

$s_1 - s_2 - s_3 - s_4 - s_5 - s_6 - s_7 - s_8 - s_9$

The first element in the set S is s_2, which is in the second position in the list. We set p_1 = 2 and we eliminate s_2 from the list:

$s_1 - \quad s_2 - s_3 - s_4 - s_5 - s_6 - s_7 - s_8 - s_9$

The second element in S is s_3, which is the second element in the remaining list, then we set p_2 = 2 and we eliminate s_3 from the list:

$s_1 - \quad s_2 - \quad s_3 - s_4 - s_5 - s_6 - s_7 - s_8 - s_9$

The process continues with s_6 that is in position 4, and then with s_8 that is in position 5 after the elimination of s_6. The resulting chromosome is [2,2,4,5]. This encoding rule allows to obtain chromosomic representations for which each entry k = 1, ..., M of the array is allowed to take values in a fixed range [1,M −k+1]. This encoding allows the construction of feasibility preserving operators as they eliminate the possibility of creating infeasible solutions after crossover and mutation operators are applied to the individuals. This means that all chromosomes obtained represent subsets with exactly M wells selected. This is an advantage of the pigeon-hole coding with respect to others, more details and examples of this encoding can be found in [22], where a similar idea is used in permutation problems. This same work shows that the phenotype expression of these solutions can be obtained in O(M logM) time. 4.2. Proposed Fitness Function The fitness function will be defined mainly as the cost. However, combi- nations of wells for which there is no feasible flow can exist. In the literature many techniques to deal with constraints in genetic algorithms have been proposed, see for example [9, 24, 28]. In this paper infeasible networks are penalised to avoid them propagating into future generations. The form of the fitness function is given by equation (23). F(S) = C $_S$fs *$^{\max\{}$1,1 + $^{H(f}$$_s$*H$^{)}$ $_{tol}$

− H$_{tol}$

}

(23)

Here, H$_{tol}$ is the maximum error that should exist between the desired and

obtained network formed concentrations, by the wells fs*in ^{is} S, ^{the} whilst ^{flow} in the same network. $^{\text{vector obtained in section 2 for the}}$ C $_S$f s *represents the cost of this flow

This fitness function takes the cost value if there is a feasible flow. In

the the opposite value of the case, objective the term function (H(f s*$^{)−H}$will

$_{tol}$)/Hincrease $_{tol}$ is ^{in} positive ^{relation} and $^{to the}$ consequently $^{cost.}$ The

last expression is and H$_{tol}$ the bigger ^{a} will ^{relative} be the ^{error,} penalty ^{the} and ^{bigger}

thus ^{the} there ^{difference} will be ^{between} an incentive $^{H(f}$ s$_{to}$ $^{*)}$ descend to combinations that provide feasible flows [28]. 4.3. Proposed Crossover and Mutation

Crossover consists in the combination of genetic material from at least two individuals (parents) in order to produce offspring. This is usually done by

splitting the chromosomic representation at a chosen point and exchanging material from both genes in order to produce two individuals (offspring). Alternatively, there have been more complex crossover operations that have been defined, for example multi-point crossover proposed by [14]. We used a variant of a multi-point crossover which allows to preserve feasible individuals after the application of the operator and not losing information in the process. In this crossover variant, the chromosomes of the parents are reordered by using a permutation π chosen at random, the permuted chromosomes are then split in a randomly selected point to then exchange the genetic material based on this point following the classical crossover operator mechanism. Finally, the two new chromosomes representing the offspring are reordered using the inverse permutation

π^{-1}. This variant was tried in [22] showing being more effective than regular multi-point crossover functions. The mutation process is very important to avoid the accelerated conver- gence and provide chances of completely exploring the feasible space. In our case, the mutation operator works by selecting an individual gene from a chromosomic representation for an individual. The selected gene is changed for other gene feasible for the current encoding, i.e., if the gene k is selected then the value at position k (denoted by p_k) is changed to any value in the range [1,M −k+1] which is the set of feasible values for the gene in position k. It also important to say that crossover and mutation are applied only to a fraction of the individuals in the current population, that fraction is a parameter of the GA and is usually defined before the algorithm is executed. There are possible ways of creating an evolving mutation pressure [11], but that is out of the scope of the present work.

5. Numerical Results

To evaluate the efficiency of the proposed methods, an instance of the problem with 90 extraction wells, 8 mixing pools, 6 evaporation pools and 10 components was simulated. The chemical qualities of the brine on each well were simulated using a normal distribution with mean μ_e and variance σ^2_e specific for each component, these distributions were taken from a real-life dataset which cannot be revealed due to confidentiality restrictions. In table 1 the values for each one of the nine components of the brine are shown, also explicit on the table are three ranges of variability for each component (Low, Medium and High). Let us recall here that the tenth component of the brine

is water, and that this component is fixed after the remaining nine compo- nents are determined in order to accomplish the desired chemical balance for the brine. Following a similar technique, the concentrations required for the product were simulated at the evaporation pools.

	K^+	Na^+	Mg^{++}	Ca^{++}	SO^{--}_4	Li^+	Cs^+	Rb^+	Cl^-
μ_e	4	6	1.5	0.05	1.6	0.2	0.002	0.002	15
σ_e (Low)	1.2	1.8	0.45	0.015	0.48	0.06	0.0006	0.0006	4.5
σ_e (Medium)	1.6	2.4	0.6	0.02	0.64	0.08	0.0008	0.0008	6
σ_e (High)	2	3	0.75	0.025	0.8	0.1	0.001	0.001	7.5

Table 1: Values used to generate concentrations

The maximum flows in the wells, minimum flows in the sinks and costs for every arc of the system were obtained from uniform distributions that were defined based on real-life examples. In table 2, the bounds for each uniform distribution used later in numerical simulations are shown.

	F^{max}_s	F^{min}_p	$c_{i,p}$	$c_{s,i}$ (Low)	$c_{s,i}$ (Medium)	$c_{s,i}$ (High)
Uniform[a, b]	[100,500]	[500,1500]	[50,300]	[50,250]	[250,750]	[750,1000]

Table 2: Range of values to generate capacities and demands

Finally, the 90 extraction wells were grouped in 9 categories depending on the range of variation of the cost of their connections and the variability σ_e with which they were simulated, see table 3.

Wells	Cost	Deviation σ_e
1-10	Low	Low
11-20	Medium	Low
21-30	High	Low
31-40	Low	Medium
41-50	Medium	Medium
51-60	High	Medium
61-70	Low	High
71-80	Medium	High
81-90	High	High

Table 3: Cost level and deviation associated to each well of the instance

The rationale for this categorisation was to try the efficiency of the GA to determine the low cost wells over the rest. Also, different deviations allow for heterogeneous wells and thus provide more chances to obtain feasible flows.

Once a set of parameters were fixed, a representative instance of a real operation was simulated, this instance being used for all the subsequent numerical experiments. All the numerical experiments were implemented in Matlab 2015b R and run over a two-cores Intel R Xeon R 2.10 GHz processor with 120 GB RAM.

5.1. Results of the Algorithm on a Fixed Network

In this subsection the results for the iterative algorithm proposed in section 3.2 are shown. In the first experiment the algorithm was run in a network formed by the first 30 wells, the first 6 mixing pools and the first 4 terminals. The ε parameter was set to 150 on each pool and the bound for the flow was set at $H_{tol} = 0.005$. Table 4 shows the detail associated with the execution of the algorithm on each iteration. It can be seen that the cost increments on each iteration in exchange for an improvement in the error H. Also, on each iteration the upper bound for cost is activated by flow, this indicates that the algorithm hasn't yet reached a local minimum for the error function H. The algorithm finally stops because the feasibility condition is satisfied on the tenth iteration because $H(f^{(10)}) \approx 0.0048 < H_{tol} = 0.005$, which corresponds to the tolerance for the tolerance parameter used.

Iteration k	Cost $C(f^{(k)})$ $10^6\times$	Chemical Feasibility Error $H(f^{(k)})$	Number of Linear Problems Solved	Time (s)	Step Size α_k	Upper Bound for Cost $(1+\alpha_k)\sigma^*$
0	1.28006	0.0387624	1	0.06792		
1	1.33824	0.0249418	5	0.33961	0.0454545	1.33824
2	1.3939	0.0202474	8	0.54338	0.0889328	1.3939
3	1.44723	0.0171903	3	0.20377	0.130599	1.44723
4	1.49843	0.014379	8	0.54336	0.170599	1.49843
5	1.54767	0.0122924	5	0.33958	0.209061	1.54767
6	1.59508	0.0103324	4	0.27168	0.246098	1.59508
7	1.64079	0.00868291	4	0.27172	0.281812	1.64079
8	1.68493	0.00770056	13	0.88299	0.316295	1.68493
9	1.7276	0.00593553	12	0.81506	0.349628	1.7276
10	1.76889	0.00484909	14	0.95091	0.381886	1.76889

Table 4: Detail of the first 10 iterations of the algorithm

The relationship between the required concentrations and the ones obtained by the algorithm solution can be observed in Table 5.

Final Concentrations Obtained by the Solution

p_i	K^+	Na^+	Mg^{++}	Ca^{++}	SO_4^{--}
1	4.21571	6.95081	0.985001	0.278726	1.68288
2	3.90522	5.78063	1.48044	0.0824016	1.59547
3	3.70371	5.59635	1.48876	0.0439441	1.55669
4	3.59542	6.24282	1.49284	0.0656936	1.5741

p_i	Li^+	Cs^+	Rb^+	Cl^-	H_2O
1	0.1935	0.00958781	0.0100856	17.6296	68.0441
2	0.243044	0.0039782	0.00737257	16.9589	69.9426
3	0.214274	0.00273388	0.00321942	15.8596	71.5307
4	0.215697	0.00280991	0.00258133	14.2589	72.5492

Expected Concentrations in Terminals

p_i	K^+	Na^+	Mg^{++}	Ca^{++}	SO_4^{--}
1	4.16501	7.23714	0.908952	0.345733	1.70645
2	3.93163	5.75791	1.46414	0.0658964	1.62662
3	3.63805	5.57221	1.57454	0.0503128	1.64443
4	3.52376	6.33129	1.68021	0.0522424	1.56423

p_i	Li^+	Cs^+	Rb^+	Cl^-	H_2O
1	0.184971	0.0069297	0.00826535	17.7987	67.6378
2	0.240016	0.0049896	0.00553624	16.7229	70.1803
3	0.180214	0.00210902	0.00167513	15.4334	71.9031
4	0.203566	0.00196312	0.00227591	12.867	73.7735

Table 5: Comparison between concentrations obtained and expected in for ten compounds

The next experiment performed was designed to answer the following question: What would happen if we change the 30 wells initially chosen?, i.e., if we chose a different set of 30 wells leaving all the other parameters equal. On the first column of Table 6 the wells chosen are individualised (out of a list of 90 wells of our previously simulated instance), the second column is the cost for the flow that is obtained in the step k = 0 of the algorithm, i.e., when the flow is minimised without considering the chemical feasibility constraint (see problem (16)). The third column of the table just shown the chemical feasibility error of the initial (unconstrained) solution. The remaining columns are concerned with the application of the iterative algorithm and show the cost, the error, number of iterations and time respectively of the application of the iterative algorithm. It is important to note the great behavioural difference that exists between problems of the same size, but for whom the only difference are the initial chemical compositions for the brines on the extraction wells. In particular, it can be seen that for the second set (wells from 11 to 40), it was not possible to attain a feasible solution, the algorithm stopped on the third iteration without finding a chemically feasible flow, i.e., the algorithm stopped because

Selected	Minimum cost, Problem (16)		Iterative Algorithm			
wells	$Cf^{(0)}$ ($10^6\times$)	$H(f^{(0)})$	Cf^* ($10^6\times$)	$H(f^*)$	Iter.	time (s)
1–30	1.2801	0.0388	1.7689	0.0048	11	5.23
11–40	1.3885	0.0727	1.3339	0.0674	3	0.38
21–50	1.1606	0.1056	1.2639	0.0050	3	8.30
31–60	1.1606	0.1056	1.3120	0.0033	4	8.14
41–70	1.0314	0.2386	1.2074	0.0048	5	35.47
51–80	1.0157	0.2192	1.1483	0.0036	4	24.87
61–90	1.0157	0.2192	1.1483	0.0038	5	25.23

Table 6: Variation of the thirty extraction wells

$Cf^{(3)} < Cf^{(0)}(1 + \alpha_3)$ (see step 3 of the algorithm in section 3.2). The fact that there are some sets of wells for which there is no chemically feasible flow justifies the choice of fitness function for the genetic algorithm (see 23). Also, it can be seen that the total cost associated to the feasible flow changes greatly depending on which 30 wells are used in the brines extraction operation; in the next section the numerical results relating to finding which 30 wells to use by means of a genetic algorithm will be discussed. Table 7 compares the performance of the proposed algorithm in relation to other established algorithms. The summary of the average obtained for the 6 problems that were run previously for which there was a chemically feasible solution is reported. For the analysis, the problem instance for which there was not chemically feasible flow, according to the tolerance parameter $H_{tol} = 0.005$, was excluded from the reported results.

	Minimum Cost (CPLEX)	Iterative Algorithm	MINOS	BARON
Cost Cf^* (multiplied by 10^6)	1.11068	1.3081	2.127	2.048
Chemical Feasibility Error $H(f^*)$	0.1545	0.0042	0.005	0.005
Solver Iterations	1	6	1413	1874
Computational Time (s)	0.48	17.87	268.24	> 300

Table 7: Comparison between minimum cost flow, iterative algorithm, MINOS and Baron

In Table 7, the first column corresponds to the solution of minimum cost without chemical specification constraints (16). The last three columns present a comparison between the solution obtained by the iterative algorithm developed in this work and the solutions obtained by commercial software such as MINOS [25] and BARON [30]. In all cases, the problem that was solved was (19) with prefixed tolerance of $H_{tol} = 0.005$, none of the two software shown results in reasonable time for the second case where the wells used were from 11 to 40. It can be observed that the minimum cost solution is far from the other solutions from a chemical concentration of the final product point of view, thus not representing a real solution to the problem. It also needs to be highlighted that each iteration of the proposed algorithm requires solving a non-linear problem, which is solved using the Frank-Wolfe method which in turn performs several iterations (see problem (10). This helps to explain the big difference that exists between the number of iterations and the computational time required to solve the problem. We are specially concerned about computational times due to the need of using the solution method as a subroutine in the genetic algorithm, the iterative algorithm is shown to be better than commercial software in both aspects, time and quality of solution. The last experiment was performed on the same instance created artificially and consisted on incrementing the network size. For this purpose, six evaporation pools and eight mixing pools were used and the number of extraction wells were incremented by 10 on each problem. The results of this experiment are shown in table 8.

Amount	Minimum cost, Problem (16)		Iterative Algorithm			
of wells	$C f^{(0)}$ $(10^6\times)$	$H(f^{(0)})$	$C f^{*}$ $(10^6\times)$	$H(f^{*})$	Iter.	time (s)
30	1.6194	0.1334	2.7764	0.0386	25	31.93
40	1.0833	0.1392	1.7678	0.0350	21	35.83
50	1.0833	0.1392	1.6995	0.0188	19	109.39
60	1.0833	0.1392	1.7896	0.0160	22	184.06
70	0.1000	0.2485	1.5631	0.0083	19	205.19
80	0.1000	0.2485	1.5586	0.0081	19	229.57
90	0.1000	0.2485	1.4418	0.0046	13	212.56

Table 8: Sensitivity to size for the proposed algorithm

The results shown in Table 8 should not be surprising as they prove that increasing the number of evaporation pools (from 4 to 6), and hence increasing the number of chemical constraints, makes it more difficult for the algorithm to find a solution. With few wells it becomes harder to satisfy

all the chemical constraints on the evaporation pools. The reader can note that as more wells are added, there are more degrees of freedom on the mixing pools and the values for the chemical error $H(f^*)$ diminishes. This observed behaviour allows to justify the operational design considerations in the mining of Lithium rich brines.

5.2. Results of the Genetic Algorithm

In this subsection the results obtained after implementing the genetic algorithm are shown. Three different tests were run iterating 20 generations with 100 individuals. In the experiments some parameters such as crossover and mutation probabilities were changed, also the number of extraction pumps and the initial population chosen. On the first execution of the GA, M = 20 wells was considered to be the size of the wells subset and a random initial population. In the second run, the number of extraction pumps was increased to M = 30 and the initial population is chosen at random again. On the third run 30 pumps were considered but the initial population was built using only wells with high and medium cost, the rationale behind this choice was to see the capabilities of the GA to eliminate costly wells and obtain individuals with good cost. Table 9 shows the probabilities used on each case.

	Run 1	Run 2	Run 3
Crossover Probability	0.8	0.8	0.9
Mutation Probability	0.1	0.1	0.2

Table 9: Crossover and Mutation Probabilities

The graph of figure 3 shows the evolution of the fitness function through 20 iterations. The dashed line represents the average fitness of all generations while the solid line shows the fitness evolution for the best individual. The horizontal line corresponds to an estimate of the best fitness, this value has been calculated evaluating the fitness of the individual possessing the 30 lowest cost simulated wells. In Table 10 the wells that are used on the GA solution for each run are presented. On each case, the solution given by the GA corresponds to the individual with better fitness found in 20 generations. Additionally, the wells in the solution are classified according to their costs (see Table 3). The row corresponding to Run 0, represents the best fitness approximation.

2
3

Figure 3: Average (segmented) and Best Fitness (continuous) for the 3 runs of the GA

It can be observed in Table 10 that the solutions are composed, mostly, by the use of low cost sources. This points out to a good performance of the genetic algorithm. Also, the fitness value for the best individual on each run are all of them relatively close to the referential cost, with the exception of the third run that obtained a higher cost. The increase in the number of wells from the first to the second run does not translates into a growth in cost, this is because the costs considered are a unit cost and the flows remain the same. On Figure 3 it can be seen that the average curve for Run 1 starts over its analogue of Run 2. The increase of the average is due to the penalty factor used in the fitness function, because by using 20 wells instead of 30 it becomes more difficult to achieve the desired concentrations and several individuals end up being infeasible ones. The average curve for Run 3 falls too quickly when compared to the other two runs, this indicates the quick elimination of the high/medium cost wells from the solution and the impact this has on the fitness function. It needs to be noted that in this last run the average curve also starts below the curve of the first run, this due to the higher number of wells and absence of penalty for the fitness. This last

Solution for Run	Low Cost Wells	Medium Cost Wells	High Cost Wells	Individual's Fitness $\times 10^5$
Run 0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 61, 62, 63 64, 65, 66, 67, 68, 69, 70			8.7358
Run 1	3, 4, 8, 31, 32, 34, 35, 36, 40, 61, 62, 63, 64, 67, 68, 70	11, 79	59, 60	8.7771
Run 2	2, 3, 4, 6, 7, 31, 33, 34, 35, 36, 37, 38, 40, 60, 61, 62, 63, 64, 65, 66, 67, 68	47, 73, 74, 80	21, 29, 86, 88	8.7464
Run 3	3, 4, 6, 31, 32, 34, 35, 36, 40, 61, 62, 63, 64, 65, 67, 68, 69	15, 41, 75, 76, 77, 79, 80	22, 23, 26, 59, 81, 84	8.8533

Table 10: Obtained solutions and their classifications

shows that the penalty scheme used is good enough to differentiate from the expensive solutions to the problem. Finally, a comparison between the fitness results of runs two and three with the results shown in the fourth column of Table 6, show the need to find a strategy that allows the planner to appropriately select the 30 wells to be used in the extraction of the brines. For example, in run two where the initial population was taken at random, the cost was 8.7464×10^5, whilst the best combination of wells in table 6 was combination 51–80 with a cost of 1.1483×10^6.

6. Conclusions

This paper studied a problem which is associated to the location of extraction pumps for the mining of Lithium, product that is more utilised nowadays. To approximate the solution for the general problem, the work was divided into two stages. On the first stage the feasibility problem with minimum cost for a fixed network was solved by using an iterative scheme based on non-linear optimisation techniques, this stage provides a solution that is able to provide a final product within specification of its chemical properties. On the second stage, the location of the best places to extract brine as to produce a product within specification requirements and minimum cost was sought, this stage utilises the methods of the first stage to evaluate the appropriateness of a given candidate solution and uses this information in the search of an optimal solution to the general problem.

The problem over a fixed network seeks a solution over bounded sections of the feasibility set. The pump location problem was modelled as a combinatorial problem and solved using a genetic algorithm to find approximate solutions to the problem. Both problems were solved on a simulated instance to show the correctness of the proposed approach and due to confidentiality issues with the real world data. It needs to be said that all problems obtained from the simulated instance are representative of a real operation. The iterative method proposed in this work has shown better feasible solutions to the problem than the one that can be obtained by commercial software such as MINOS and BARON. In addition, the computational required by the iterative method also showed a better behavior, which allowed us to use this method to define the fitness function of the Genetic Algorithm, even though a chemically feasible flow could not be found for some configurations of fixed networks. The Genetic Algorithm has shown to be useful in finding solutions that use wells that provide flows with the expected quality and with a good cost. On the Table 10, it can be seen that the GA is able to identify and maintain in the population pool those solutions the low cost sources included in the instance which suggests a correct implementation and performance. The difficulty of this method lie on the higher computational requirement as for each individual of population (network) the fitness function requires the resolution of a problem over a fixed network. Despite this increase in computational time, the proposed GA is appropriate to solve the extraction planning problem for Lithium deposits as this problem does not need to be solved too frequently.

Acknowledgment

The authors wish to thanks to the anonymous referees for their careful readings and constructive criticism which provide a substantial improvement of the paper.

References

[1] Adhya, N., Tawarmalani, M., Sahinidis, N. V., 1999. A lagrangian approach to the pooling problem. Industrial & Engineering Chemistry Research 38 (5), 1956–1972.

[2] Alfaki, M., Haugland, D., 2011. Comparison of discrete and continuous models for the pooling problem. In: OASIcs-OpenAccess Series in Informatics. Vol. 20. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[3] Almutairi, H., Elhedhli, S., 2009. A new lagrangean approach to the pooling problem. Journal of Global Optimization 45 (2), 237–257.

[4] Audet, C., Brimberg, J., Hansen, P., Digabel, S. L., Mladenovic, N., 2004. Pooling problem: Alternate formulations and solution methods. Management science 50 (6), 761–776.

[5] Baker, T. E., Lasdon, L. S., 1985. Successive linear programming at exxon. Management science 31 (3), 264–274.

[6] Bazaraa, M. S., Sherali, H. D., Shetty, C. M., 2013. Nonlinear programming: theory and algorithms. John Wiley & Sons.

[7] Ben-Tal, A., Eiger, G., Gershovitz, V., 1994. Global minimization by reducing the duality gap. Mathematical programming 63 (1), 193–212.

[8] Bertsekas, D. P., 1999. Nonlinear programming. Athena scientific Belmont.

[9] Coello, C. A. C., 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer methods in applied mechanics and engineering 191 (11), 1245–1287.

[10] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., et al., 2001. Introduction to algorithms. Vol. 2. MIT press Cambridge.

[11] Eiben, Á. E., Hinterding, R., Michalewicz, Z., 1999. Parameter control in evolutionary algorithms. Evolutionary Computation, IEEE Transactions on 3 (2), 124–141.

[12] Floudas, C. A., Aggarwal, A., 1990. A decomposition strategy for global optimum search in the pooling problem. ORSA Journal on Computing 2 (3), 225–235.

[13] Foulds, L., Haugland, D., Jörnsten, K., 1992. A bilinear approach to the pooling problem. Optimization 24 (1-2), 165–180.

[14] Frantz, D. R., 1972. Non-linearities in genetic adaptive search.

[15] Garrett, D. E., 2004. Handbook of lithium and natural calcium chloride. Academic Press.

[16] Gonzalez, A., 2000. Riquezas minerales de chile a nivel mundial.

[17] Gupte, A., Ahmed, S., Dey, S. S., Cheon, M. S., 2013. Pooling problems: relaxations and discretizations. School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA. and ExxonMobil Research and Engineering Company, Annandale, NJ.

[18] Gupte, A., Ahmed, S., Dey, S. S., Cheon, M. S., 2015. Relaxations and discretizations for the pooling problem. Journal of Global Optimization, 1–39.

[19] Haverly, C. A., 1978. Studies of the behavior of recursion for the pooling problem. Acm sigmap bulletin (25), 19–28.

[20] Holland, J. H., 1975. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

[21] Kallrath, J., 2000. Mixed integer optimization in the chemical process industry: Experience, potential and future perspectives. Chemical Engineering Research and Design 78 (6), 809–822.

[22] Lam, S. S., 1996. Genetic algorithm with pigeon-hole coding scheme for solving sequencing problems. Applied artificial intelligence 10 (3), 239–256.

[23] Meyer, C. A., Floudas, C. A., 2006. Global optimization of a combinatorially complex generalized pooling problem. AIChE journal 52 (3), 1027–1037.

[24] Michalewicz, Z., Janikow, C. Z., 1991. Handling constraints in genetic algorithms. In: ICGA. pp. 151–157.

[25] Murtagh, B. A., Saunders, M. A., 1983. Minos 5.0 user's guide. Tech. rep., DTIC Document.

[26] Pham, V., Laird, C., El-Halwagi, M., 2009. Convex hull discretization approach to the global optimization of pooling problems. Industrial & Engineering Chemistry Research 48 (4), 1973–1979.

[27] Quesada, I., Grossmann, I. E., 1995. Global optimization of bilinear process networks with multicomponent flows. Computers & Chemical Engineering 19 (12), 1219–1242.

[28] Richardson, J. T., Palmer, M. R., Liepins, G. E., Hilliard, M., 1989. Some guidelines for genetic algorithms with penalty functions. In: Proceedings of the third international conference on Genetic algorithms. Morgan Kaufmann Publishers Inc., pp. 191–197.

[29] Ruiz, M., Briant, O., Clochard, J.-M., Penz, B., 2013. Large-scale standard pooling problems with constrained pools and fixed demands. Journal of Global Optimization 56 (3), 939–956.

[30] Sahinidis, N. V., 1996. Baron: A general purpose global optimization software package. Journal of global optimization 8 (2), 201–205.

[31] Sarker, R. A., Gunn, E. A., 1997. A simple slp algorithm for solving a class of nonlinear programs. European Journal of Operational Research 101 (1), 140–154.

[32] Tawarmalani, M., Sahinidis, N. V., 2002. Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications. Vol. 65. Springer Science & Business Media.

[33] Visweswaran, V., Floudast, C., 1990. A global optimization algorithm (gop) for certain classes of nonconvex nlpsii. application of theory and test problems. Computers & chemical engineering 14 (12), 1419–1434.