Journal of Cleaner Production 144 (2017) 149–160

Contents lists available at ScienceDirect

Journal of Cleaner Production

journal homepage: www.elsevier.com/locate/jclepro

A comparison between ACO and Dijkstra algorithms for optimal ore concentrate pipeline routing

Daniel Baeza [a, b, *], Christian F. Ihle [c, b], Julián M. Ortiz [a, c, 1]

[a] Advanced Laboratory for Geostatistical Supercomputing (ALGES), University of Chile, Tupper, 2069, Santiago, Chile
[b] Advanced Mining Technology Center (AMTC), University of Chile, Avda. Tupper 2007, Edificio AMTC, Santiago, Chile
[c] Department of Mining Engineering, University of Chile, Avda. Tupper, 2069, Santiago, Chile

ARTICLE INFO

Article history:
Received 8 March 2016
Received in revised form
16 December 2016
Accepted 17 December 2016
Available online 20 December 2016

Keywords:
Pipeline
Ant Colony Optimization
Operations research
Combinatorial optimization
Slurry

ABSTRACT

One of the important aspects pertaining the mining industry is the use of territory. This is especially important when part of the operations are meant to cross regions outside the boundaries of mines or processing plants. In Chile and other countries there are many long distance pipelines (carrying water, ore concentrate or tailings), connecting locations dozens of kilometers apart. In this paper, the focus is placed on a methodological comparison between two different implementations of the lowest cost route for this kind of system. One is Ant Colony Optimization (ACO), a metaheuristic approach belonging to the particle swarm family of algorithms, and the other one is the widely used Dijkstra method. Although both methods converge to solutions in reasonable time, ACO can yield slightly suboptimal paths; however, it offers the potential to find good solutions to some problems that might be prohibitive using the Dijkstra approach in cases where the cost function must be dyamically calculated. The two optimization approaches are compared in terms of their computational cost and accuracy in a routing problem including costs for the length and local slopes of the route. In particular, penalizing routes with either steep slopes in the direction of the trajectory or high cross-slopes yields to optimal routes that depart from traditional shortest path solutions. The accuracy of using ACO in this kind of setting, compared to Dijkstra, are discussed.

1. Introduction

Today, dozens of slurry pipelines longer than 100 km are running in the world. They mostly serve the mining, energy and dredging industries (Abulnaga, 2002), and feature a solid and liquid phase, the latter being the vehicle to transport the former. In Chile, most of these correspond to concentrate or tailings pipelines, and a significant part operates near either agricultural or populated areas. In most countries, including Chile, it is required to submit an environmental impact assessment to the environmental authority. However, this is done after the formulation of the project, where crucial aspects such as the route have been already decided to maximize operational smoothness, as commented in the context of bauxite and copper concentrate transport in Gandhi et al. (2006) and Betinol and Navarro (2009), respectively. The lack of concurrency between project and stakeholder feedback and expectations precludes, in general, the opportunity to look at an optimal solution for the long distance pipeline design problem in a broad sense. This can be phrased equivalently as to look at the shortest possible pipeline given a set of constraints (Dey, 2002), to cope with a myriad of restrictions. In particular, there is a number of elements combining socio-environmental, operational and investment aspects that define, in addition to the route, the way the system should be operated not only for cost minimization, but also to potentially promote cleaner production practices including the identification of safer routes and lower carbon footprint operational points. Integrated frameworks using decision support systems have been used to analyze this kind of infrastructure problems.

In particular, the analytic hierarchy process (AHP), developed by Saaty in the seventies (Bhushan and Rai, 2004, and references therein) has been extensively used. Here, hierarchy assignment, prioritization and calculation steps need to be applied. After a set of

* Corresponding author. Advanced Laboratory for Geostatistical Supercomputing (ALGES), University of Chile, Tupper, 2069, Santiago, Chile.
E-mail address: dabaeza@alges.cl (D. Baeza).
[1] Present address: Robert M. Buchan Department of Mining, Queen's University, Kingston, Ontario K7L 3N6, Canada.

http://dx.doi.org/10.1016/j.jclepro.2016.12.084
0959-6526/

desirable characteristics the outcomes of the project should have, the AHP methodology is used to assign a weight to both objective and subjective elements such as length, operability, maintainability, potential of failures and impacts to the communities during planning, construction and operation. A critical aspect of this decision-making algorithm is the need to defining a finite set of possibilities to select from. It is thus required to count on an expert advise to pre-select a few of them out of a potentially large amount of possibilities. For instance, following this procedure as a case study of a pipeline project in India, Dey (2002, 2006) have obtained optimal routes between 6% and 7% longer than the shortest possible ones but with total NPV values between 4% and 5% lower than those corresponding to the shortest route.

The use of decision support systems to identify a finite number of routing options is a convenient choice in grid-like environments where the number of possible paths is relatively straightforward to identify. An example is the computation of lowest cost paths in urban routing problems of water pipelines (Simpson et al., 1994; Luettinger and Clark, 2005), where blocks and the existing infrastructure become hard restrictions to the routing. Differently from such configurations, cross-country pipelines offer enormous amounts of possibilities, and thus are best tackled with combinatorial optimization schemes. In cross-country pipelines, the routing options are given by the possibilities offered by the topography: civil works allow for the construction on virtually any possible hilly condition, bringing a feasibility problem to a cost one. However, many applications can still be found where the decision is made based on an expert's assessment of feasible routes, instead of through combinatorial optimization. This usual approach to pipeline routing is shown for example in Fookes et al. (2001) in Algeria, where land type classification is performed as a first step to later determine by expert knowledge possible paths for the pipeline, without an exhaustive determination of the best route. The paper focusses mostly in the important subject of land characterization for the purpose of costing and constraining the route. Humber and Eng (2004) focus on pipeline route selection (field based versus desk based) through proper data collection, dynamic routing consideration and the use of information from multiple sources. They do not clearly state an approach for finding the optimum path and they rely on a combination of expert knowledge supported by GIS technologies for handling and utilizing the field data available.

Additionally, routing decisions will also depend on the fluid or slurry being conveyed in the pipeline. In particular, an important difference between water and slurry pipelines is the impact of high amounts of solids —commonly exceeding 28% by volume in mineral processing operations (Abulnaga, 2002)—, which causes that their route is additionally constrained by flow-related features such as slope- and slurry-dependent energy consumption along with solids accumulation at bottom sections (Wilson et al., 2006). Also, operational- and planning-related issues including the implications of the pipeline route on system restarts after recent shutdowns (Ihle, 2013, 2014) are major issues to consider when a solid phase with the potential of settling is present. Different routes impact the selection of critical equipment including the number of pumps, the characteristics of energy dissipation stations (often required to keep pressure above the vapor value in the system), the pipeline diameter, its overall length and the operational sequence. In particular, higher average slopes and the presence of low points tend to reduce the maximum allowable shutdown times while the system is laden with solids.

Equally important is the final routing and the potential impact of leaks on the environment, as it is a source of conflict between the pipeline operator, communities and the environmental authorities (Ihle et al., 2014). Thus, a natural related issue is to solve the problem of minimizing environmental damage and, at the same time coping with production goals, instead of posing the *ex-post* question regarding the effect of an already designed or existing pipeline. The solution to the problem of the best route for a slurry cross-country pipeline is far from being a one-size-fits-all type. Although in common engineering practice, the route is chosen by an experienced engineer based on capital and operational cost considerations (Gandhi et al., 2006), similarly as in submarine pipelines (de Lucena et al., 2014), so far there is not a quantitative tool able to handle the general problem of optimizing a generalized definition of capital, operational, environmental and social cost of pipelines, explicitly including the hundreds or even thousands of routing possibilities.

Research has been published to conceptualize the routing of cross-country pipelines discretizing the topography as a grid. In the oil and gas sector, Shamir (1971) posed a route optimization problem considering both infrastructure and the energy consumption through a measure of the pressure difference that the pumping system would need to supply. To this purpose, he proposed a dynamic programming algorithm to obtain the least cost route in a territory corridor. Here, operational costs have been interpreted in light of the energy required for pumping. Middleton and Bielicki (2009) proposed an integrated algorithm featuring a modified version of the Dijkstra approach (Dijkstra, 1959) for carbon capture and storage network planning. In their set of constraints they considered sensitive areas as part of the criteria to find the best pipeline array. More recently, Marcoulaki et al. (2012) have studied the problem of pipeline route optimization in terms of the effect of route perturbations on infrastructure, operation and maintenance costs, from a predefined set of options. The same group of authors later extended their single-objective-function framework to a broader space of pipeline configurations to include the effect of corrosion (Marcoulaki et al., 2014). In both cases, they use simulated annealing for the solution of the iterative optimization problem. Applications to oil and gas are similar, but constraints are linked to slope stability and the associated risk of failure of the pipe, or physical obstacles to the route, due to other existing pipelines or some other submarine obstacles. de Lucena et al. (2014) discusses an application of genetic algorithms to the selection of submarine pipeline routes for oil and gas, considering aspects related to the parametrization of possible routes, implementation of these constraints in the genetic algorithm (see also Fernandes et al., 2009), and the definition of the objective function that incorporates these constraints. Constraint-handling techniques has also been the focus of other publications (Mezura-Montes and Coello, 2011; Takahama et al., 2005). Genetic algorithms and other nature inspired algorithms have also been developed for these kinds of applications. In particular, Vieira et al. (2008) develop an optimization based on artificial immune systems, while de Pina et al. (2011) develop a particle swarm optimization.

The work of de Lucena et al. (2014) indicates that selection has been traditionally manually performed. It also defines hard and soft constraints, that make the solution infeasible in the first case, or inconvenient in the second. It should be pointed out that in our case, these criteria can be input either by eliminating areas from the solution space where ants can move, or by weighting their cost in the cost function. Interesting techniques are presented to handle and determine the weight assigned to each constraint.

The present work is within the framework of an ongoing project devoted to integrating the aforementioned cost dimensions to the pipeline design problem including route operation and environmental impact. The objective is two-fold. First, to show that when including the topography as a cost element, the optimal routing may differ from the shortest one, thus moving away from the paradigm ‘shorter is better’. Second, to compare the results of the

present implementation, using an ant colony optimization approach (also referred to herein as ACO, Dorigo, 1992), with an exhaustive method, namely Dijkstra, and thence to show its suitability for tackling least cost problems applied to pipeline routing that exhaustive algorithms might not handle properly in some situations. In particular, in a broad pipeline routing optimization framework the Dijkstra approach may become unsuitable. Examples are when tunnels or bridges may be required and, on the other hand, when curvature restrictions on the construction need to be fulfilled. In the first case, the cost function is not a direct result of the distance, but indicator variables must be added to the decision rule (namely, building or not a tunnel or bridge). This implies that the possibilities to be explored grow exponentially, making the exploration of all solutions unfeasible in practice. On the other hand, the curvature in long distance pipelines represents a limitation to the constructibility. It is common practice in long distance cross-country pipeline construction to avoid the use of elbows to change the direction of the route in 45° or 90° when required, thus avoiding accelerated wear and enhanced hydraulic energy losses due to the presence of such singular points. Instead, the pipeline is slightly bent in the field, creating smooth changes in direction. From the point of view of the least cost problem implementation, this implies the need to define a cost function that is dynamically computed, which depends on the various feasible routing possibilities (Dorigo et al., 2006) to account for a curvature restriction. This dynamic definition cannot be implemented using Dijkstra but is, however easy to implement in ACO.

Differently from the Dijkstra algorithm, the ACO approach is not exhaustive and reaching the global optimum is not always ensured (Gutjahr, 2000, 2002; Stutzle and Dorigo, 2002). Furthermore, even when convergence is guaranteed, the speed at which it occurs may be too slow for practical purposes, hence a potentially suboptimal, albeit good solution is often accepted in a reasonable timeframe for practical applications.

By analogy with the construction of motorways, building slurry pipelines on hilly topographies requires, directly or indirectly, to decide on the slope distribution along and perpendicular to the trajectory. While the former impacts the ease of transportability of the slurry (Wilson et al., 2006), the latter has direct implications on the constructibility and the cost required to build the pipeline. In particular, the volume of material required to be removed is an increasing function of the cross slope. In the present paper, the effect of the relation between the route and the topography, as a pre-requisite to the implementation of a more complete framework for slurry pipeline optimization, is analyzed.

2. Optimization algorithm description

Long distance pipeline routes span distances that may reach several hundred kilometers (see Jacobs, 1991; Abulnaga, 2002; for reviews), and relevant changes on the topography, including slope, curvature, diameter, the presence of an intermediate pumping and/or energy dissipation station, tunnels, etc., might occur in distances as small as dozens meters. Therefore, when facing a node-arc scheme when setting a minimum cost problem, it is not uncommon to require the definition of more of than one thousand nodes and arcs in two-dimensional routes and perhaps millions of them in three-dimensional problems when tunnels and bridges are considered *a priori*. To date, the most common approach to solve shortest path problems is Dijkstra or variants such as the A* algorithm.

The Dijkstra method, which progressively selects the least cost route by comparing among all the possibilities ensures, by construction, an optimal solution. However, its computational cost on the order of $O(N^2)$, or $O(E + N\log N)$, with N and E the number of nodes and edges, respectively (Fredman and Tarjan, 1987), may easily become prohibitively high in a multi-kilometer routing problem. Furthermore, during this process, the cost function needs to be already fixed to allow evaluation and subsequent comparison. The A* search algorithm is an extension of Dijkstra that optimizes the search by choosing paths that are estimated to be shorter through an auxiliary function that provides a bound of the actual cost (Zeng and Church, 2009). Although the A* algorithm is in many cases more efficient than Dijkstra, we have decided to consider Dijkstra as our base case, since both algorithms suffer similar limitations in terms of computational cost and flexibility to consider a dynamically calculated cost function, while Dijkstra is more widely known.

Among the most popular metaheuristic frameworks to solve problems with large combinatorial solution spaces are those based on genetic algorithms, simulated annealing, ant colony optimization, iterated local search, and tabu search. A good taxonomy of these methods and their description can be found in Trabelsi et al. (2010). In this section, we describe the exhaustive approach provided by Dijkstra's algorithm and the alternative metaheuristic of ant colony optimization, which is suited to solve this problem and has the potential of handling additional complexities in the cost function.

2.1. Dijkstra's algorithm

The shortest path problem, as described in Korte and Vygen (2007), is defined by the minimum distance between two nodes over a connected graph and a set of edges with conservative and non-negative weights. Dijkstra provides a way to find the optimal path between two points in a graph. Algorithm 1 in Appendix A shows a schematic description of Dijkstra logic and Fig. A.12 shows the meaning of each parameter of Dijkstra in the real problem.

2.2. Ant colony optimization method

The ACO algorithm (Dorigo, 1992; Dorigo et al., 2006) is inspired on the ant colony behavior to find food and take it to the nest, where there is a random component on the trajectory of the ants looking for food prior to finding it. When successful in finding the food, the ants deposit a pheromone on the ground, which biases the path decision of other ants of the colony as a hint to obtain food from the same source. The evolution of the routes to reach an optimum (shortest or safest route) is performed by combining the effect of adding pheromone when successful ants return to the nest and wearing of the pheromone as a function of time. As ants use previously visited paths to the food, the stronger the pheromone footprint is, the more successful ants are to come back to the nest with the food. The imposed random walk to increase the probability of finding food along with the balance between pheromone decay and enhancement in various routes allows the colony to change paths until convergence is obtained. Differently from Dijkstra, ACO does not look at every possible route, and thus may converge to a sub-optimal solution. However, for large scale problems, ACO could be a better option than Dijkstra, considering that it is not an exhaustive method and does not need to evaluate all the possible paths.

The method is implemented considering the following steps:

1. Initialization: relevant parameters are set for the subsequent iterations.
   (a) Reading of surface: $z_{ij} = f(x_{ij}, y_{ij})$.

(b) Defining the cost function parameters: here, three parameters $\tilde{c}_0 > 0$, $\tilde{c}_1 > 0$, $\tilde{c}_2 > 0$, are considered to account for the length of the pipeline, and the costs related to the slope (parallel to the trajectory of the pipeline) $p_{1,k}$, which is linked to maximum shutdown times, and to the cross slope $p_{2,k}$, which has implications during the construction and maintenance requirements.

(c) Constructing a graph to compute the cost of moving from a node to any of the eight adjacent ones in the regular lattice. For each arc in the graph, a cost is associated.

(d) Defining the size of the ant colony and the number of iterations of the algorithm.

2. Iterations: each iteration represents one time step, where ants move from their current location to a new one. Initially all ants leave the nest.

(a) Move from current location one step, that is, to one of the allowable adjacent nodes of the graph. Higher pheromone arcs have a higher probability of being used.

i. If an ant has already found food, it will go back to the nest, taking the successful route and leaving pheromones to notify other ants.

ii. If an ant has not found food yet, a new arch is randomly chosen with a probability that depends on the pheromone footprint. The probability that the ant k takes the edge (i,j) is given by the following equation:

$$Prob_{i,j}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{w} \tau_{iw}^{\alpha} \eta_{iw}^{\beta}} & \text{if } (ij) \text{ is a feasible edge} \\ 0 & \text{if } (ij) \text{ is not a feasible edge,} \end{cases} \quad (1)$$

where τ_{ij} is the pheromone available at the edge (ij), α is a free parameter, η_{ij} is the inverse of the edge cost and β is a free parameter. The parameters α and β are adjustable and control the relative importance between the pheromone variable and the heuristic information, i.e., the cost function value for each edge.

Additionally, to get a better convergence, free parameter Q_0 is introduced. For each iteration, the ants can choose between following the best possible edge or other (sub-optimal) feasible edges. This is done through a random number q with uniform distribution between 0 and 1. This number is generated in every decision taken for each ant. If q is greater than Q_0, then the ant chooses the best possible edge. If q is lower than Q_0, the ant follows the edge obtained by Equation (1). This allows to move away from local minima and increase the chances of exploring a larger set of possible solutions.

(b) Update pheromone footprint:

i. If the ant has already found food and is returning to the nest, pheromone is added to the arc.

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + Agn_{t_{back}}^{k} \cdot \delta\tau_{ij}^{k}, \quad (2)$$

where ρ is a free parameter related with the evaporation of the footprint, between 0 and 1, and $Agn_{t_{back}}^{k}$ is an aging parameter. The latter is used to impose the pheromone footprint decay as it approaches to the nest, marking more intensely the nodes close to the food. The parameter $\delta\tau_{ij}^{k}$ is the pheromone contribution for the ant k in the edge ij.

ii. If the ant has not found food, the pheromone is wore out of the arc by a given amount. For every graph edge (i,j) the pheromone will be reduced at a rate of ϕ as:

$$\tau_{ij} \leftarrow (1-\phi)\cdot\tau_{ij} + \phi\cdot\tau_0, \quad (3)$$

where τ_0 is the pheromone of the edge i,j and corresponds to the minimum value that the pheromone can take and ϕ is a free parameter related to the wearing of the pheromone footprint produced by the passage of the ant through the edge ij.

(c) Stopping criteria:

i. If the maximum number of iterations has been reached, stop and keep the current route as optimum, provided the food has been found.

ii. If not, continue.

Algorithm 2 in Appendix B shows a schematic description of the ant colony optimization algorithm. Fig. B.13, in the same appendix, shows a complementary flowchart.

2.3. Implementation

In the present work, the solution and a visual interface, to display the colony behavior and the pheromone evolution in time have been implemented in C code and the open source visualization toolkit OpenFrameworks (Perevalov, 2013). This allows assessing different cost function components and their consequences in the final optimum path.

3. Problem description

3.1. Cost function

A cost function based on parallel and cross-slope components is considered as in Baeza et al. (2015). A (x_{ij}, y_{ij}, z_{ij}) mapping is used to allow for potential routes, discretized by the index sets i_k and j_k $(k \varepsilon \{0,\ldots,N-1\})$. The elevation corresponding to the topography is mapped as a scalar function of (x_{ij}, y_{ij}) as $z_{ij} = f(x_{ij}, y_{ij})$. The pipeline is allowed to take any possible triad set (x,y,z). The slope parallel to the trajectory affects the slurry flow both during operation and also after system shutdown (Ihle, 2014). Fig. 1 shows a schematic of the comparative effect of a hilly and a flat topography on the parallel slope. In the hilly case, depicted in Fig. 1a, the ups and downs of the topography create differences between the particle and the fluid velocity during the operation, affecting operational costs (Doron et al., 1997) and, on the other hand, may induce the formation of solid plugs at low points when the pipeline is shut

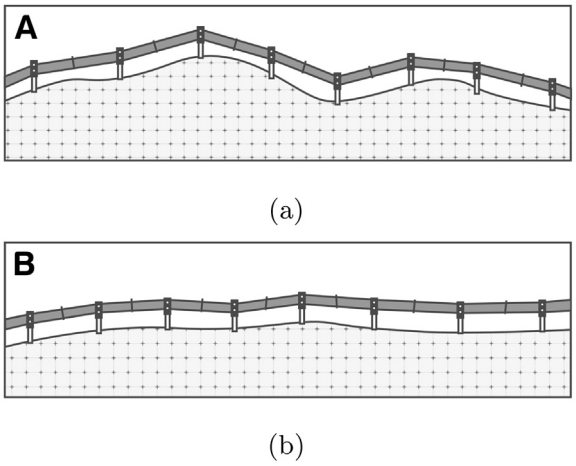


Fig. 1. A comparison between parallel slopes: (a) hilly topography and (b) flat topography. Parallel slopes have significant implications for operation of the pipeline.

down (Ihle, 2014). In contrast, the flat topography (Fig. 1b) is neutral from the perspective of the particle buoyancy effect on the axial direction of the flow and, upon system shutdown, leaves a particle-free cross section that enables a smooth restarting, triggered by a top-down particle resuspension mechanism. Although it is beyond the scope of the present work to use a model to account for the complexities of the particle migration mechanism in a resuspension model, a cost component proportional to the parallel slope is considered.

A schematic of the cross-slope component is shown in Fig. 2, both for the effect of a high slope and nil cross-slope (Fig. 2a and b, respectively). It is shown that, in the first case, a considerable amount of material needs to be removed in comparison to the case of Fig. 2b, with the obvious implications in capital costs and system constructibility. Other less evident implications are the risk of failure due to landslides, caused by seismicity (Keefer, 2000) or rain (Iverson, 2000), and the potential for limited accessibility for maintenance purposes, depending on the route characteristics.

The aforementioned elements configure a non-trivial problem for the pipeline route. The optimization problem to be solved is:

$$\Omega^* = \min_{i,j} \Omega \tag{4}$$

$$\Omega = \mathbf{C} \cdot \mathbf{I}, \tag{5}$$

where the dot represents the inner product operation between two vectors and (i,j) are possible sequences of indices of the pipeline trajectory. The components of the arc vector, l_k, are $||x_k - x_{k-1}||$, with $k \in \{0, \ldots, N-1\}$. The components of the cost function vector, $\mathbf{C}$, are given in terms of the various arcs connecting the nodes of a given path as $\tilde{C}_k = \tilde{c}_0 + \tilde{c}_1 p_{1,k} + \tilde{c}_2 p_{2,k}$, with $k \in 0, \ldots, N-1$. Dividing the cost function by $\tilde{c}_0 > 0$ the following dimensionless cost function is obtained:

$$C_k = 1 + c_1 p_{1,k} + c_2 p_{2,k} \quad k \in 0, \ldots, N-1, \tag{6}$$

where the dimensionless cost components c_1 and c_2 are positive. The dimensionless components $p_{1,k}$ and $p_{2,k}$ represent the parallel and the cross-slopes, respectively, for a given path, as described in Appendix C.

It is noted that, if c_1 and c_2 are zero, the problem reduces to the shortest path problem, since the cost to be minimized is just the total length, whose unit cost is c_0.

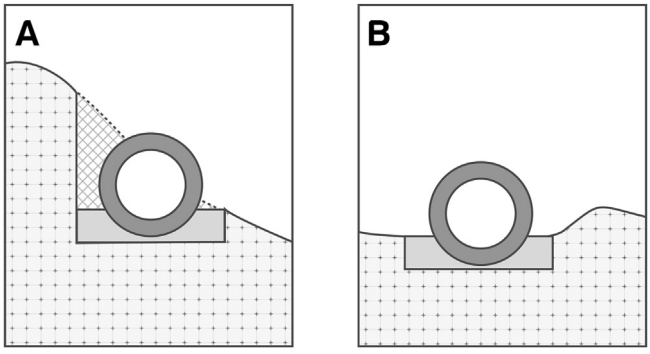


Fig. 2. A comparison between cross slopes: (a) high slope and (b) nil slope. Cross slope has significant implications during construction and maintenance.

3.2. Topographies

To test the accuracy and convergence of ACO in this particular kind of problem, besides the components of the cost function as a comparing element, the topography plays, in principle, a major role on the assessment of the present heuristic method. In particular, while Dijkstra considers every possible path regardless how intricate is the topography, this is not the case when using an algorithm based on a particle swarm approach, where convergence is rather given by a particular organization of the population.

In the present work, a Gaussian obstacle (Fig. 3a) and a natural topography have been considered. The latter has been obtained from a digital elevation model of a image extracted from a catalog of NASA images. This area is located in the southeast of San Martin de Los Andes, Argentina and its name is Corral de Piedra (Fig. 3b). While the effect of the cost components is fairly simple to anticipate in the Gaussian obstacle, it is not so in the natural topography example, as expected in many real-world applications.

4. Results and discussion

4.1. The relevance of the cost components

The dimensionless cost components c_1 and c_2 may combine in different ways to signify the relative impact of the cost on the specific path in terms of the slope parallel and perpendicular to the trajectory. In particular, the matrix of combinations shown in

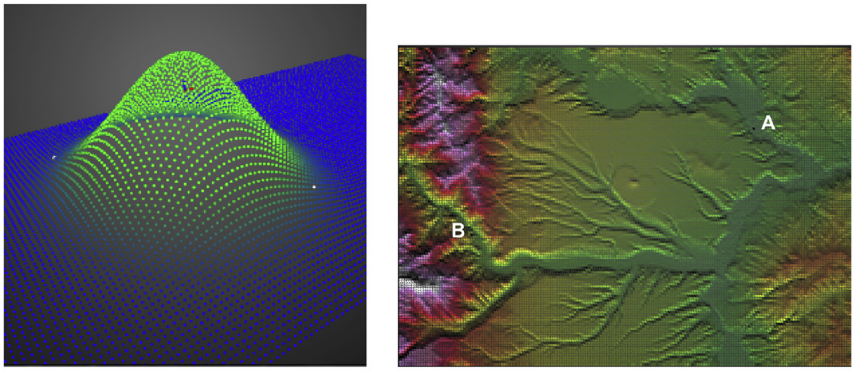


Fig. 3. Topographies considered in the analysis. (a) Gaussian hill, (b) natural topography, obtained from a digital elevation model. A and B are the starting and ending point, respectively.

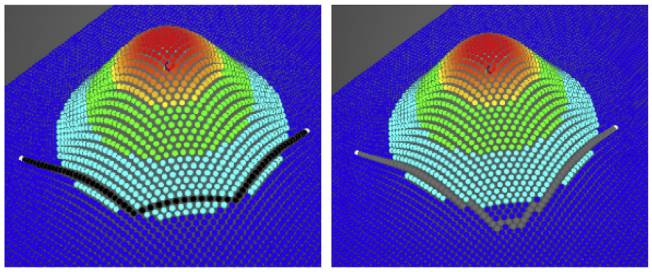
Table 1
Combination matrix for $c_j = \bar{c}_j/\bar{c}_0$, with $j \in \{1,2\}$ to be used in both the Gaussian obstacle an the natural topography.

	Case 1	Case 2	Case 3	Case 4
c_1	0	0	2	2
c_2	0	2	0	1

Table 1, with values of c_1 and c_2 between 0 and 2, will be analyzed herein, for both the Gaussian obstacle and the natural topography.

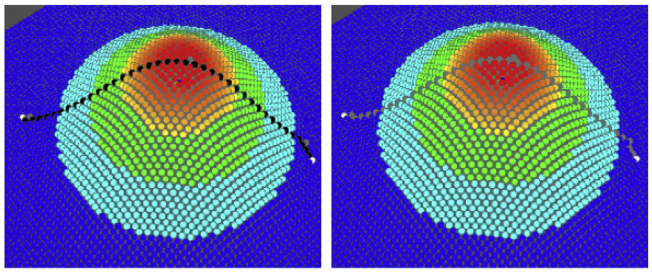
When the parallel and perpendicular slope components are 0, i.e., $c_1 = c_2 = 0$ (case 1G), the only relevant driver of the cost function is the cost of the Euclidean norm of the path or, equivalently, the cost vector **C** such as $C_k = 1$ for $k \in \{1, \ldots, N\}$. The optimal solution for the best path starting on a nearly horizontal location and ending on a symmetrical location is given by Fig. 4. This optimal solution is significantly different to that obtained when the parallel slope cost is important compared to that of the cross slope. Fig. 5 shows the optimal solution for $c_1 = 0$ and $c_2 = 2$ (case 2G). This result is somewhat more intuitive: as there is no incentive for taking paths with a cross-slope component, the best route is the shortest possible with no cross-slope component, with the total cost corresponding to the sum of the effect of the parallel slope and the overall length. If the start and the end points are symmetric to the origin, then the solution is a route whose plan projection is a straight line. In real systems this route may prove unfeasible if the parallel slopes are too high (*e.g.* precluding vehicle inspection if too high).

In contrast with case 2G, shown in Fig. 5, where $c_1/c_2 = 0$, in case 3G (Fig. 6b) $c_2/c_1 = 0$, which will make parallel slopes a comparatively expensive option. The corresponding outcome is a route through relatively low altitudes, running through the skirt of the hill.



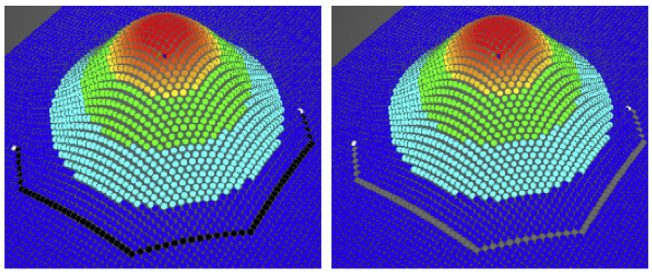
(a) Route value = 0.102 (b) Route Value = 0.116

Fig. 4. Optimal solution, corresponding to $c_1 = c_2 = 0$ (case 1G). (a) Route obtained using the Dijkstra algorithm (b) Route obtained using the ACO algorithm.



(a) Route value = 0.113 (b) Route value = 0.116

Fig. 5. Least cost path, corresponding to $c_1 = 0$ and $c_2 = 2$ (case 2G). (a) Route obtained using the Dijkstra algorithm (b) Route obtained using the ACO algorithm.



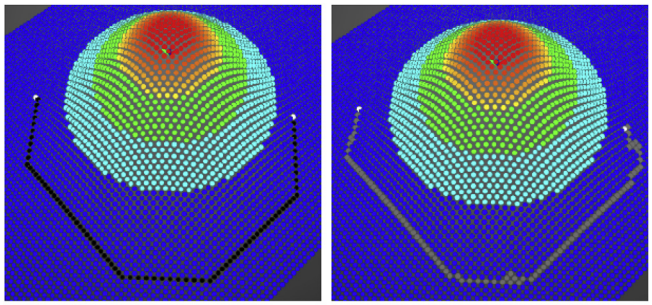
(a) Route value = 0.158 (b) Route value = 0.158

Fig. 6. Least cost path, corresponding to $c_1 = 2$ and $c_2 = 0$ (case 3G). (a) Route obtained using the Dijkstra algorithm (b) Route obtained using the ACO algorithm.

A less obvious optimal route corresponds to non-trivial combinations of the cost dimensions, as in case 4G. Here, there is a competition between the cost components given by $c_1 = 2$ and $c_2 = 1$. Differently from case 2G ($c_1 = 0$ and $c_2 = 2$), there is a positive cost for routing in the direction of the slope. However, the cost due to cross-slopes along the route somewhat pushes the routes away from the summit, leading to the use of the skirt of the hill instead of the stronger parallel-slope section. Fig. 7 clearly shows that this increase in the cross-slope component makes more attractive a longer option compared to case 3G ($c_1 = 2$ and $c_2 = 0$).

The examples shown above show that accounting for additional cost components leads to switching the problem from a classic shortest path problem to a minimum cost one. The resulting costs will change depending on the unit costs assigned to the parallel and cross slopes, as shown in Table 2.

In slurry pipelines, the dimensionless cost vector **C** may be interpreted in terms of potentially feasible values of the parallel and cross slope components $p_{1,k}$ and $p_{2,k}$, respectively. In the case of $p_{2,k}$, there are no restrictions to the values it may take, and may thus exceed the unity if slopes greater than 45° are cut in hills. Thus, restrictions often come due to higher earthwork costs for



(a) Route value = 0.191 (b) Route value = 0.202

Fig. 7. Least cost path, corresponding to $c_1 = 2$ and $c_2 = 1$ (case 4G). (a) Route obtained using the Dijkstra algorithm (b) Route obtained using the ACO algorithm.

Table 2
Optimal path costs and associated lengths for the Gaussian obstacle, obtained from each case (arbitrary units).

	Case 1G	Case 2G	Case 3G	Case 4G
ACO costs	0.116	0.116	0.158	0.202
Dijkstra costs	0.102	0.113	0.156	0.191
ACO number of nodes	87	52	86	104
Dijkstra number of nodes	74	50	85	99

maintenance track construction, associated to higher cross-slopes. In the parallel slope component a common practice is to limit $p_{1,k}$ to about 0.18.

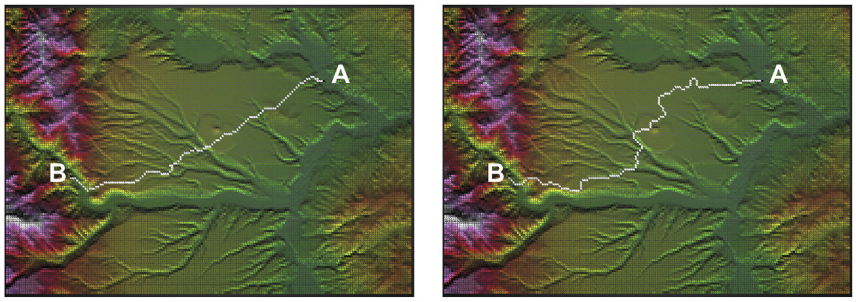
4.2. Natural topography

In this case, the importance of each cost function component is tested over a natural topography, represented by a graph of 135×192 nodes with 407573 edges. In addition, Dijkstra is used to measure the accuracy of the solutions given by ACO.

The first experiment only uses a Euclidean cost component, that is, $c_1 = c_2 = 0$. The result of this experiment is shown in Fig. 8. In this case, Dijkstra and ACO found routes that go through the shortest path connecting A and B, regardless the presence of valleys and hills.

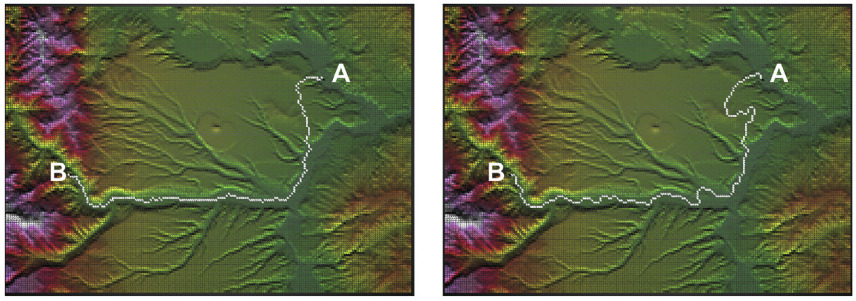
The second experiment exposes the relevance of the parallel cost component. The cost function is configured as $c_1 = 2$ and $c_2 = 0$ for this case. Fig. 9 shows the impact of the parallel cost over the least cost route. In this case Dijkstra and ACO try to avoid steep slopes, taking a significant part of the route through a naturally formed plateau between A and B. The final portion of the path is almost perpendicular to the plateau section, which is the shortest, albeit predominantly flat section along the route. This route goes, however near the skirt of the same hill range, implying the lack of importance of the cross slope.

The next case shows the impact of the cross slope component of the cost function, set as $c_1 = 0$ and $c_2 = 2$. Fig. 10 shows the best route founded by Dijkstra and ACO algorithms. Although the choice of the best route found using ACO and Dijkstra shares with that shown in Fig. 9 a significant portion on top of the plateau, the second section is developed differently. In particular, the route through the skirt of the hill range is avoided and and s-shaped path on the flat section of the terrain is preferred. This choice implies a longer route but, overall, the lowest cost.



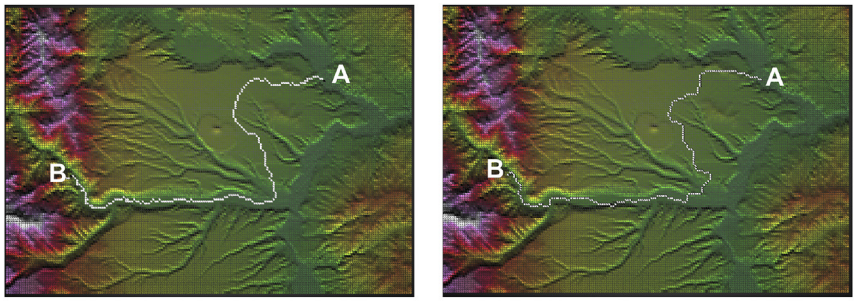
(a) Dijkstra (b) ACO

Fig. 8. Least cost path over a natural topography, corresponding to $c_1 = c_2 = 0$. (a) Route obtained using the Dijkstra algorithm (b) Route obtained using the ACO algorithm.



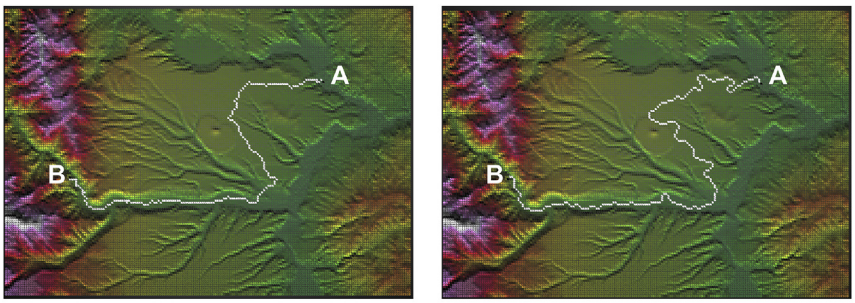
(a) Dijkstra (b) ACO

Fig. 9. Least cost path over a natural topography, corresponding to $c_1 = 2$ and $c_2 = 0$. (a) Route obtained using the Dijkstra algorithm (b) Route obtained using the ACO algorithm.



(a) Dijkstra (b) ACO

Fig. 10. Least cost path over a natural topography, corresponding to $c_1 = 0$, $c_2 = 2$. (a) Route obtained using the Dijkstra algorithm (b) Route obtained using the ACO algorithm.



(a) Dijkstra (b) ACO

Fig. 11. Least cost path over a natural topography, corresponding to $c_1 = 2$ and $c_2 = 1$. (a) Route obtained using the Dijkstra algorithm (b) Route obtained using the ACO algorithm.

Finally, we consider the case when the three components of the cost function are assumed non-zero, $c_1 = 2$ and $c_2 = 1$, implying that both the parallel and cross-sectional components of the cost function are equally important. As in the previous cases, both Dijkstra and ACO methods found similar routes (Fig. 11a and b, respectively) with a small difference near the end of the route. Comparing with the shortest path problem (Fig. 8), the optimal solution in this case was considerably different, as it took most of the route on top of the topography plateau.

4.2.1. Comments on the ACO solution accuracy

In general, convergence between ACO and the optimal solution requires a few thousand iterations of the algorithm. This is equivalent to refer to a few thousand configurations of the ants position and the pheromone field. In spite of the apparently large time required for convergence, in the present system, containing on the order of 25,000 nodes, it was required around a minute to achieve convergence. It is noted, however, that using the same machine (Intel Xeon E5, 3.3 GHz CPU frequency, 10 MB, 16 MB RAM and 1 TB HDD) and solving the same problem in a grid of the same size, Dijkstra required considerably less computational effort, it is virtually instantaneous (less than 0.5 s). The point of testing ACO in this section is not to look for a more efficient way to solve the present problem, but to test its accuracy in this kind of problem. Table 3 summarizes the differences between ACO and the (optimal) Dijkstra solution in the natural topography case.

5. Conclusions

Several factors need to be considered to define a good pipeline route. Construction, maintenance and operational costs are among the main issues in the evaluation of this kind of project. In this work, a cost function has been presented to consider these aspects for optimization purposes, understanding a broad sense of the variables to be optimized. It is clear from the present results that adding such additional elements would not necessarily yield the shortest path, but is likely to yield a better one, in the sense of an ad-hoc cost formulation. In the present, simple approach, the definition of the cost function considers, in addition to the infrastructure, a cost proportional to the route length, penalizing functions for along-slope and cross-slope routes through the cost components c_1 and c_2. The intent of such a definition is precisely to expose the implications of operations and constructibility in route planning, and the need to put together operation and engineering in a unified approach. The cost function has been tested using the Dijkstra algorithm, widely used for this kind of problem, which is able to find the shortest path between two points, and ACO. Although in the present set of simulations Dijkstra was faster and more accurate than ACO, it has been proven in the context of the simulations that ACO yields very reasonable results. The real advantage of ACO does not actually come with the present application, but one step ahead of it, where a dynamic cost function would be required to consider and when the solution is required in larger problems. This kind of problem would add a degree of complexity and computational cost that would deem unfeasible the use of Dijkstra. Importantly, this will make optimization viable in large scale, real pipeline routing problems. A mature pipeline routing system including operation, capital costs, environmental and communitary aspects, and thus embracing the concept of cleaner production, would encourage the improvement of a regulatory framework for long distance pipeline routing, where the best practice will be implemented in the form of a set of cost restrictions (or even hard restrictions when required). Such regulations, to be imprinted on an optimization routine, will also create debate and involvement of operators and stakeholders on the definition of a broader cost function. On the other hand, an open methodology for pipeline cost optimization will enable replication and improvement in time. From this point of view, the present research is also devoted to implementation of a reasonably accurate, highly flexible computational approach, based on known principles, to put together designers, operators and communities in front of local authorities to set routing rules. In particular, both the decisions that a consultant needs to make to solve for a specific pipeline route and how to weight a specific cost dimension have in common the need to define a criterion. An optimization approach like this one offers the possibility to make it objective and repeatable. This is, perhaps, its most powerful feature.

Table 3
Summary of the convergence using ACO, compared to Dijkstra, for the case of the natural topography.

	Case 1N	Case 2N	Case 3N	Case 4N
ACO costs	0.43	3.98	5.53	15.63
Dijkstra costs	0.36	3.68	5.12	15.47
ACO number of nodes	159	205	241	241
Dijkstra number of nodes	128	189	210	200

Acknowledgements

The author gratefully acknowledge support from the Department of Mining Engineering of University of Chile. CFI gratefully acknowledges support from ALGES Laboratory and the Chilean National Commission for Scientific and Technological Research, CONICYT, through Fondecyt Project no. 1160971.

Appendix A. Dijkstra algorithm

Input: Graph bidirected G with weights $c : E(G) \to \Re_+$ and a vertex $s \in V(G)$

Output: Lengths of shortest path from s to all $v \in V(G)$. All $l(v)$ and $p(v)$ for all $v \in V(G)$ are calculated. Where $l(v)$ is the length of a shortest $s - v - path$. If v is not accessible from s, then $l(v) = \infty$ and the edge $p(v)$ is undefined.

1 Set $l(s) := 0$. Set $l(v) := \infty$ for all $v \in V(G)\backslash\{s\}$. Set $R := \emptyset$;
2 Find a vertex $v \in V(G)\backslash R$ such that $l(v) = \min_{\omega \in V(G)\backslash R} l(\omega)$;
3 Set $R := R \cup \{v\}$;
4 **for** *all* $\omega \in V(G)\backslash R$ *such that* $(v, \omega) \in E(G)$ **do**
5 **if** $l(\omega) > l(v) + c((v, \omega))$ **then**
6 Set $l(\omega)) := l(v) + c((v, \omega))$ and $p(\omega) := v$;
7 **end**
8 **end**
9 **if** $R \neq V(G)$ **then**
10 **go to** line 2
11 **end**

Algorithm 1: Dijkstra's algorithm. Transcription from Korte and Vygen (2007). A visual explanation of the algorithm is shown in Fig. A.12.

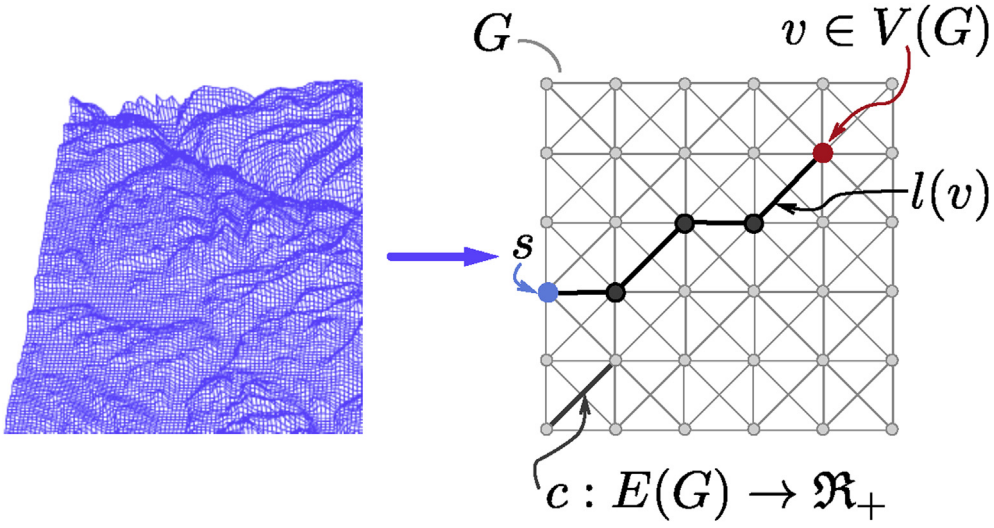


Fig. A.12: Visual explanation of notations in Dijkstra and ACO.

Appendix B. Details of ACO algorithm and its implementation

Algorithm 2 shows the ACO algorithm used in the present set of runs. Fig. B.13 shows the flow chart that summarizes the present computational implementation.

Input: A digraph G, weights $c : E(G) \to \Re_+$, *Nest* node index and *Food* node index.

Output: Shortest path from *Nest* node to *Food* found by the colony after achieved the stop criteria.

```
Set pheromone parameters;
Set Nest node and Food node;
while !stopCriteriaIsReached do
    for all Ants of the Colony do
        if !GoBackToTheNest then
            SelectFeasibleEdges();
            ChooseNextNode();
            MoveToTheNextNode();
            if foodFound then
                UpdateBestRoute();
            end
        else
            MoveBackToTheNest();
        end
        UpdatePheromone();
    end
end
```

Algorithm 2: ACO algorithm.

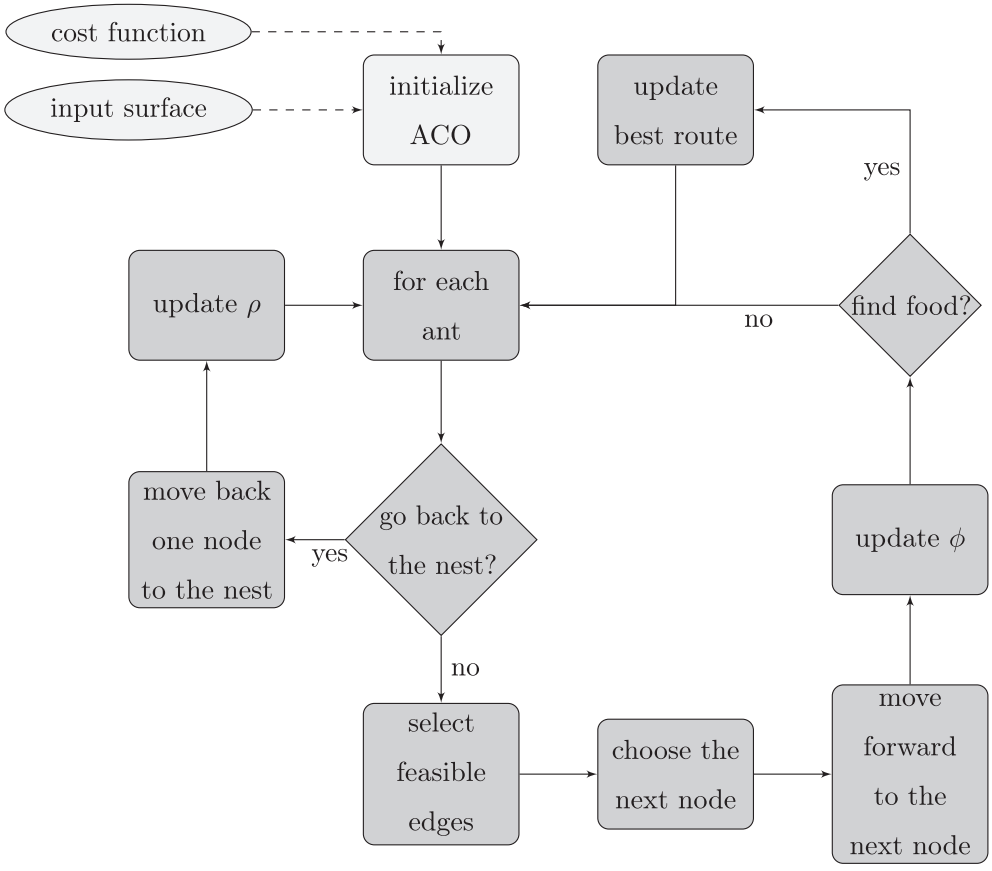


Fig. B.13: Flow chart of ACO implementation.

Appendix C. Parallel and cross slope components

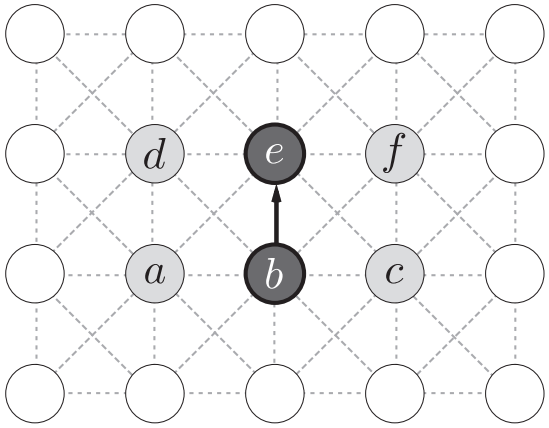
To compute the parallel and cross-slope components ($p_{1,k}$ and $p_{2,k}$, respectively), two cases (labelled herein as a and b) have to be considered. The first, labelled herein as case a, corresponds to the computation of the components when the trajectory is parallel to the grid while in the second (named to herein as case b) the trajectory is diagonal. Both are depicted in Fig. C.14 and they are defined as:

$$p_{1,k} = \begin{cases} \dfrac{c_z - b_z}{\sqrt{(c_x - b_x)^2 + (c_y - b_y)^2}} & \text{(a)} \\ \dfrac{f_z - b_z}{\sqrt{(f_x - b_x)^2 + \left(f_y - b_y\right)^2}} & \text{(b).} \end{cases} \tag{C.1}$$

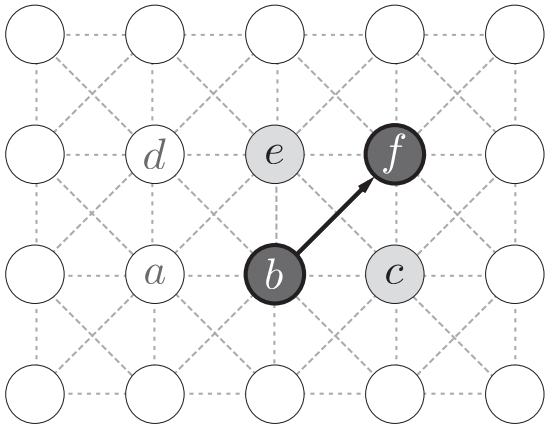
On the other hand, the cross-slope, $p_{2,k}$, is expressed as:

$$p_{2,k} = \begin{cases} \dfrac{\dfrac{d_z + a_z}{2} - \dfrac{f_z + c_z}{2}}{\sqrt{\left(\dfrac{d_x + a_x}{2} - \dfrac{f_x + c_x}{2}\right)^2 + \left(\dfrac{d_y + a_y}{2} - \dfrac{f_y + c_y}{2}\right)^2}} & \text{(a)} \\ \dfrac{e_z - c_z}{\sqrt{(e_x - c_x)^2 + (e_y - c_y)^2}} & \text{(b).} \end{cases} \tag{C.2}$$

The definitions for a to f are given in Fig. C.14.



(a) Parallel to the grid



(b) Moving diagonally in the grid

Fig. C.14: Slope calculations. The black nodes are used to compute the parallel slope and the gray nodes are used to compute the perpendicular slope.

References

Abulnaga, B.E., 2002. Slurry Systems Handbook. McGraw-Hill, New York.

Baeza, D., Ihle, C.F., Ortiz, J.M., 2015. An ant colony optimization-based approach for long distance ore pipeline routing. In: Proceedings of the 37th International Symposium on the Applications of Computers and Operations Research in the Mineral Industry (APCOM). Fairbanks, Alaska.

Betinol, R.G., Navarro, L., 2009. Slurry pipeline design approach. In: Proc. Rio Pipeline Conf.

Bhushan, N., Rai, K., 2004. Strategic Decision Making: Applying the Analytic Hierarchy Process. Springer Science & Business Media.

de Lucena, R.R., Baioco, J.S., de Lima, B.S.L.P., Albrecht, C.H., Jacob, B.P., 2014. Optimal design of submarine pipeline routes by genetic algorithm with different constraint handling techniques. Adv. Eng. Softw. 76, 110–124.

de Pina, A.A., Albrecht, C.H., de Lima, B.S.L.P., Jacob, B.P., 2011. Tailoring the particle swarm optimization algorithm for the design of offshore oil production risers. Optim. Eng. 12 (1–2), 215–235.

Dey, P.K., 2002. An integrated assessment model for cross-country pipelines. Environ. Impact Assess. Rev. 22 (6), 703–721.

Dey, P.K., 2006. Integrated project evaluation and selection using multiple-attribute decision-making technique. Int. J. Prod. Econ. 103 (1), 90–103.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271.

Dorigo, M., 1992. Optimization, Learning and Natural Algorithms. Ph. D. Thesis. Politecnico di Milano, Italy.

Dorigo, M., Birattari, M., Stützle, T., 2006. Ant colony optimization. Computational intelligence magazine. IEEE 1 (4), 28–39.

Doron, P., Simkhis, M., Barnea, D., 1997. Flow of solid-liquid mixtures in inclined pipes. Int. J. Multiph. Flow 23 (2), 313–323.

Fernandes, D.H., Jacob, B.P., Lima, B.S.L.P., Medeiros, A.R., Albrecht, C.H., 2009. A proposal of multi-objective function for submarine rigid pipelines route optimization via evolutionary algorithms. In: Proc. Rio Pipeline Conf. Rio Janeiro, Brazil.

Fookes, P.G., Lee, E.M., Sweeney, M., 2001. Pipeline route selection and ground characterization, Algeria. Geol. Soc. Lond. Eng. Geol. Spec. Publ. 18 (1), 115–121.

Fredman, M.L., Tarjan, R.E., 1987. Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM (JACM) 34 (3), 596–615.

Gandhi, R.L., Norwood, J., Che, Y., 2006. Cross-country bauxite slurry transportation. In: Galloway, T.J. (Ed.), Light Met. The Minerals, Metals & Materials Society, pp. 35–39.

Gutjahr, W.J., 2000. A graph-based ant system and its convergence. Future Gener. Comput. Syst. 16 (8), 873–888.

Gutjahr, W.J., 2002. Aco algorithms with guaranteed convergence to the optimal solution. Inf. Process. Lett. 82 (3), 145–153.

Humber, J., Eng, B., 2004. Enhancing the pipeline route selection process. In: GITA Oil Gas Conf. Proc. Citeseer.

Ihle, C.F., 2013. A cost perspective for long distance ore pipeline water and energy utilization. Part I: optimal base values. Int. J. Min. Process 122, 1–12.

Ihle, C.F., 2014. Should maximum pressures in ore pipelines be computed out of system startups or power outages? Miner. Eng. 55, 57–59.

Ihle, C.F., Tamburrino, A., Montserrat, S., 2014. Identifying the relative importance of energy and water costs in hydraulic transport systems through a combined physics-and cost-based indicator. J. Clean. Prod. 84, 589–596.

Iverson, R.M., 2000. Landslide triggering by rain infiltration. Water Resour. Res. 36 (7), 1897–1910.

Jacobs, B.E., 1991. Design of Slurry Transport Systems. Routledge.

Keefer, D.K., 2000. Statistical analysis of an earthquake-induced landslide distribution–the 1989 Loma Prieta, California event. Eng. Geol. 58 (3), 231–249.

Korte, B., Vygen, J., 2007. Combinatorial Optimization: Theory and Algorithms, fourth ed. Springer Publishing Company (Incorporated).

Luettinger, J., Clark, T., May 2005. Geographic information system-based pipeline route selection process. J. Water Resour. Plan. Manag. 131 (3), 193–200.

Marcoulaki, E., Tsoutsias, A., Papazoglou, I.A., 2014. Design of optimal pipeline systems using internal corrosion models and GIS tools. Ind. Eng. Chem. Res. 53, 11755–11765.

Marcoulaki, E.C., Papazoglou, I.A., Pixopoulou, N., 2012. Integrated framework for the design of pipeline systems using stochastic optimisation and GIS tools. Chem. Eng. Res. Des. 90 (12), 2209–2222.

Mezura-Montes, E., Coello, C.A.C., 2011. Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm Evol. Comput. 1 (4), 173–194.

Middleton, R.S., Bielicki, J.M., 2009. A scalable infrastructure model for carbon capture and storage: SimCCS. Energy Policy 37 (3), 1052–1060.

Perevalov, D., 2013. Mastering OpenFrameworks: Creative Coding Demystified. Community Experience Distilled. Packt Publishing.

Shamir, U., 1971. Optimal route for pipelines in two-phase flow. Soc. Pet. Eng. J. 11 (03), 215–222.

Simpson, A.R., Dandy, G.C., Murphy, L.J., 1994. Genetic algorithms compared to other techniques for pipe optimization. J. Water Res. Plan. Manag. 120 (4), 423–443.

Stutzle, T., Dorigo, M., 2002. A short convergence proof for a class of ant colony optimization algorithms. IEEE Trans. Evol. Comput. 6 (4), 358–365.

Takahama, T., Sakai, S., Iwane, N., 2005. Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm. AI 2005 Adv. Artif. Intell. 389–400.

Trabelsi, K., Sevaux, M., Coussy, P., Rossi, A., Sörensen, K., 2010. Metaheuristics. In: Advanced Metaheuristics for High-level Synthesis. Springer, Berlin, Ch.
Vieira, I.N., de Lima, B.S.L.P., Jacob, B.P., 2008. Optimization of steel catenary risers for offshore oil production using artificial immune system. In: In: Int. Conf. Artif. Immune Syst. Springer, pp. 254–265.
Wilson, K.C., Addie, G.R., Sellgren, A., Clift, R., 2006. Slurry Transport Using Centrifugal Pumps. Chapman and Hall, London.
Zeng, W., Church, R.L., 2009. Finding shortest paths on real road networks: the case for A*. Int. J. Geogr. Inf. Sci. 23 (4), 531–543.