A Pipe Routing Hybrid Approach Based on A-Star Search and Linear Programming

Marvin Stanczak, Cédric Pralet, Vincent Vidal, Vincent Baudoui

A Pipe Routing Hybrid Approach Based on A-Star Search and Linear Programming

Marvin Stanczak[1,2(✉)], Cédric Pralet[1(✉)], Vincent Vidal[1(✉)], and Vincent Baudoui[2(✉)]

[1] ONERA/DTIS, Université de Toulouse, 31055 Toulouse, France
{marvin.stanczak,cedric.pralet,vincent.vidal}@onera.fr
[2] Airbus Defence and Space, Toulouse, France
{marvin.stanczak,vincent.baudoui}@airbus.com

Abstract. In this work, we consider a pipe routing problem encountered in the space industry to design waveguides used in telecommunication satellites. The goal is to connect an input configuration to an output configuration by using a pipe composed of a succession of straight sections and bends. The pipe is routed within a 3D continuous space divided into non-regular convex cells in order to take obstacles into account. Our objective is to consider several non-standard features such as dealing with a set of bends restricted to a bend catalog that can contain both orthogonal and non-orthogonal bends, or with pipes that have a rectangular section, which makes the pipe orientation important. An approach is introduced to automate the design of such pipe systems and help designers to manage the high number of constraints involved. Basically, this approach formulates the pipe routing problem as a shortest path problem in the space of routing plans, where a routing plan is specified by the parts composing a pipe and by geometrical constraints imposed on these parts. The problem is then solved using weighted A* search combined with a linear program that quickly evaluates the feasibility and the cost of a routing plan. The paper shows that the approach obtained has the capacity to solve realistic instances inspired by industrial problems.

Keywords: Pipe routing · Linear programming · Weighted A*

1 Introduction

Waveguide routing plays an important role during the design phase of a telecommunication satellite. A waveguide can be seen as a pipe with a rectangular section which carries an electromagnetic signal between two components of the satellite payload. Traditionally, waveguide designers define routes manually and build a route as a succession of rigid straight sections and rigid bends using their expertise to select a routing configuration that saves length and bends. The bends that can be used are defined in catalogs provided by waveguide manufacturers. These catalogs can include orthogonal bends (90° bends, when the direction of the pipe before the bend is orthogonal to the direction of the pipe after the bend)

P. J. Stuckey (Ed.): CPAIOR 2021, LNCS 12735, pp. 179–195, 2021.
https://doi.org/10.1007/978-3-030-78230-6_12

but also non orthogonal ones (30° bends, 45° bends, 60° bends, etc.) which are useful to save length in the very constrained routable space inside a satellite.

Due to the large number of waveguides to be routed and to the large number of constraints to take into account, waveguide routing is usually a highly time-consuming task. In other industries, research has been conducted to automate the pipe routing process, to save time and money during the design phase.

Existing Methods. Historically, the classic pipe routing algorithm is the *Maze algorithm* introduced by Lee [23]. It considers a discrete routing environment represented as a regular grid where the cells occupied by obstacles are labeled. Then, starting from the source cell, the best route is computed by exploring the cells' neighborhood until the target is reached. Other algorithms were proposed to consume less memory space, based on Particle Swarm Optimization [3], Genetic Algorithm [18,21], or Ant Colony Optimization [9,19]. Recently, Belov also introduced a Constraint Programming formulation [4,5]. With regard to non orthogonal bends, Ando proposed to introduce additional edges in the cell adjacency graph [2], but this approach does not guarantee to find a feasible solution even if one exists in the continuous space.

Other methods called *Skeleton approaches* build a graph of possible route candidates using rules inspired by experience. The pipe routing problem is still modeled as a shortest path problem in a discrete graph, but this time the nodes of the graph are associated with continuous positions that are not forced to be located on a regular grid. The skeleton approach was tackled using the Dikjstra algorithm [20] or Mixed Linear Integer Programming [13]. A more recent paper also explored Evolutionary Algorithms considering multi-objective routing in a skeleton graph [24]. Similarly to the cell decomposition strategy, the skeleton approach can lead to sub-optimal solutions in the continuous space.

To improve the quality of the solutions, some authors proposed a two-stage approach that first searches for a global routing channel, and then details the routes inside each cell of this channel [31], sometimes using patterns [26].

To avoid discretizing the routing space, other methods discretize the set of possible physical pipe components and build pipes by adding components one by one from a catalog containing bends and straight sections of fixed length [12]. Again, this method can fail to find feasible solutions due to the discretization.

Another way to route pipes in a continuous environment is the *Escape Algorithm* or *Line Search Algorithm.* This technique introduced by Hightower [15] extends lines from the origin point using a set of directions and repeats this extension each time a line intersects an obstacle, until the destination is reached. This approach allows dealing with non orthogonal bends but it does not take into account the orientation of the pipe sections, which is required in the case of waveguide routing. Indeed, the waveguide must reach the destination with the right orientation, that is to say with the right angular position of the section around the *neutral fibre* followed by the barycentre of the section along the pipe.

Last, parametric models were also proposed using adaptable pipe patterns with parameters such as the length of straight sections. The pipe route can then be optimized using Mathematical Programming [25,28] or Genetic Algorithms [17], but existing models do not consider non orthogonal bends.

Contribution. Most of the previous methods rely on the classical assumptions that the pipe has a circular section, has axis-parallel segments and/or uses orthogonal bends only, but these hypotheses are not acceptable for waveguides. For these reasons, no method from the literature can be reused to solve the waveguide routing problem. This article introduces a new pipe routing technique that addresses the three following challenges: optimize pipe cost in a 3D continuous routing space, use non orthogonal bends from a catalog and take into account unsymmetrical pipe sections. We consider the routing of a single pipe only. In an industrial context, such a routing problem must be solved in few seconds in order to ensure quick iterations during the design phase. The extended problem which deals with several pipes is left for future work.

The rest of the article is organized as follows. Section 2 introduces a formal definition of the pipe routing problem considered. Section 3 describes the notion of routing plan and presents a linear program that evaluates the feasibility and cost of a plan. Section 4 formulates the pipe routing problem as a shortest path problem in the space of routing plans and defines heuristics for estimating the cost required to reach the goal configuration from the current routing plan. Section 5 provides experimental results obtained with the weighted A* search, and Sect. 6 gives future work perspectives.

2 Problem Definition

2.1 Routing Space

The routing space $\mathcal{W} \subseteq \mathbb{R}^3$ describes the physical space that the pipe neutral fibre can cross. The structure of a telecommunication satellite is made of several panels on which waveguides must be fixed. Thus, the routable space is naturally split into several cells that correspond to the close environment of each panel. As about one hundred obstacles can be fixed on a panel, these cells are divided into sub-cells that avoid obstacles, using a Delaunay constrained triangulation. This way, we model the routing space $\mathcal{W}$ as a set of cells $\mathcal{C}$ where each cell $c \in \mathcal{C}$ is a convex polyhedron $\mathcal{P}_c$ (see Fig. 1).

Two cells $c, c' \in \mathcal{C}$ that overlap are called adjacent, and their *interface* i is the non-empty polyhedron $\mathcal{P}_i = \mathcal{P}_c \cap \mathcal{P}_{c'}$. We denote by $\mathcal{I}$ the set of routing interfaces between cells, and by $\mathcal{I}(c)$ the set of interfaces involving cell c. We assume here that each interface is a surface, even if the techniques defined can be extended to the case where interfaces are volumes. From these definitions, the routing space can be represented as a graph containing one node per routing cell and one edge per routing interface between two cells. In the following, this graph is assumed to be connected.

2.2 Input and Output Configurations

In pipe routing, the goal is to connect an input configuration with an output configuration. Mathematically speaking, a *configuration* θ is a pair (P_θ, o_θ) where

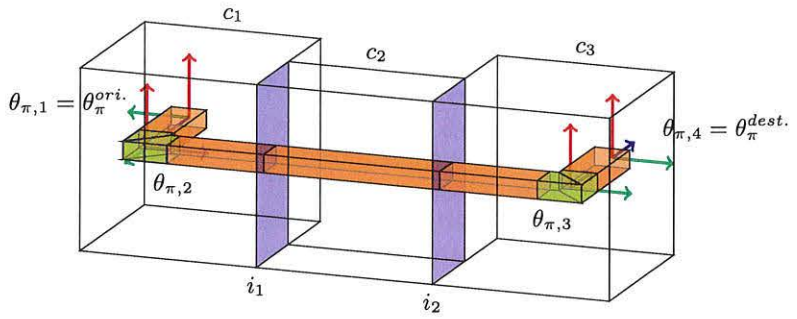

Fig. 1. Pipe in a simple routing space containing three cells (interfaces between cells are depicted in blue, straight sections in orange and bends in green) (Color figure online)

P_θ is a point in $\mathbb{R}^3$ and o_θ is an orientation defined by three vectors $(\overrightarrow{e_{o,1}}, ..., \overrightarrow{e_{o,3}})$ that form an orthonormal basis of $\mathbb{R}^3$ with $\overrightarrow{e_{o,3}}$ normal to the pipe section.

The origin point is located in a convex polyhedron $\mathcal{P}^{ori.}$ included in an origin cell $c^{ori.} \in \mathcal{C}$ and the destination point is located in a convex polyhedron $\mathcal{P}^{dest.}$ included in a destination cell $c^{dest.} \in \mathcal{C}$. Using polyhedrons $\mathcal{P}^{ori.}$ and $\mathcal{P}^{dest.}$ instead of fixed origin and destination points allows keeping some flexibility in early design phases. For instance, if a straight pipe section can connect an origin and a destination configuration up to a small position error, it can be relevant to slightly move the origin or destination point instead of defining a pipe that contains a loop to compensate for the position error.

Additionally, the (unique) origin orientation $o^{ori.}$ can be set as the reference orientation, and there exists a set of possible destination orientations $\mathcal{O}^{dest.}$, either because the destination orientation is still flexible, or because some orientations are equivalent from the pipe section point of view.

2.3 Straight Sections and Bends

To connect input and ouput configurations, engineers use catalogs of parts referred to as *straight sections* and *bends*. In our modeling, these parts are approximated as functions that transform a configuration $\theta = (P_\theta, o_\theta)$ into another configuration $\theta' = (P_{\theta'}, o_{\theta'})$.

A straight section u of length $L_u \in \mathbb{R}^+$ is modeled as an application that transforms a configuration $\theta = (P_\theta, o_\theta)$ into a configuration $\theta' = u(\theta) = (P_{\theta'}, o_{\theta'})$ where point $P_{\theta'}$ is obtained by a translation of P_θ along direction $\overrightarrow{e_{o_\theta,3}}$ ($P_{\theta'} = P_\theta + L_u \overrightarrow{e_{o_\theta,3}}$) and where the orientation is unchanged ($o_{\theta'} = o_\theta$). In the following, a translation of length L is referred to as t_L, therefore $u(\theta) = t_{L_u}(\theta)$.

The modeling of bends is a bit more complex. Basically, a bend b has a certain length and changes the direction of the pipe according to a given bend radius. It is approximated as a pipe section composed of a straight section of length $L_b \in \mathbb{R}^+$ called the *half-length* of the bend, an orientation change defined by a

rotation matrix M_b, and another straight section of length L_b again (see Fig. 1). More formally, a bend b is modeled as a function that tranforms a configuration $\theta = (P_\theta, o_\theta)$ into a configuration $t_{L_b} \circ r_{M_b} \circ t_{L_b}(\theta)$ obtained by applying (1) a translation t_{L_b} of length L_b, (2) a rotation r_{M_b} defined by matrix M_b, and (3) a second translation t_{L_b} of length L_b. For a given bend, the intermediate configuration $r_{M_b} \circ t_{L_b}(\theta)$ is called the *transition configuration*. It corresponds to the configuration obtained just after the break point of the bend.

In the following, we consider a restricted set of bends B called the *catalog of bends*. It can contain only 90° bends for instance, or 90° and 45° bends, or any other combination of bend types and angles. Using a catalog of bends can seem counter-intuitive at first since bend suppliers often propose a continuous choice of bend angle radius, and since complex bends could be obtained through 3D-printing. However, using a catalog of bends allows reusing bends and ordering numerous bends of the same type, which reduces the costs.

2.4 Pipe and Polyline Approximation

Given the catalog of bends B, a pipe π of length $N_\pi > 0$ is then a pair $\pi = (\theta_\pi^{ori.}, \sigma_\pi)$ composed of an initial configuration $\theta_\pi^{ori.}$ and a composition

$$\sigma_\pi = u_{\pi,N_\pi} \circ b_{\pi,N_\pi-1} \circ u_{\pi,N_\pi-1} \circ ... \circ u_{\pi,2} \circ b_{\pi,1} \circ u_{\pi,1}$$

alternating between straight sections $u_{\pi,i}$, for $i \in [\![1, N_\pi]\!]$, and bends $b_{\pi,i} \in B$, for $i \in [\![1, N_\pi - 1]\!]$ (see Fig. 1).

As straight sections and bends apply translation and rotation operations on the initial configuration, the neutral fibre of pipe π describes a polyline $[P_1, ..., P_{N_\pi+1}]$ in $\mathbb{R}^3$ whose points correspond to the transition configurations of the bends (except for the first and last points). The i^{th} point of this polyline is $P_{\pi,i}$, and the i^{th} segment is $[P_{\pi,i}, P_{\pi,i+1}]$. We also denote by $\ell_{\pi,i}$ the length of the i^{th} segment of pipe π, and by $\theta_{\pi,i} = (P_{\pi,i}, o_{\pi,i})$ the i^{th} transition configuration of π, for $i \in [\![1, N_\pi + 1]\!]$ (with the convention that $\theta_{\pi,1} = \theta_\pi^{ori.}$, and that $\theta_{\pi,N_\pi+1}$ is the configuration obtained at the end of the pipe).

2.5 Constraints

We now list the constraints that must be satisfied by a pipe π.

- The initial and final configurations of π must be consistent with the required input and output configurations, that means respectively $P_{\pi,1} \in \mathcal{P}^{ori.}$ and $P_{\pi,N_\pi+1} \in \mathcal{P}^{dest.}$, but also $o_{\pi,1} = o^{ori.}$ and $o_{\pi,N_\pi+1} \in \mathcal{O}^{dest.}$.
- The polyline defined by the pipe must contain at most N^{max} segments. This limits the number of break points, which is particularly useful when the pipe conveys a flow whose quality is impacted by such break points.
- Every bend used in π must belong to catalog B.
- All segments of the polyline traversed by the neutral fibre of the pipe must be contained within the routing space, that is $[P_{\pi,i}, P_{\pi,i+1}] \subset \mathcal{W}$ for every

$i \in [\![1, N_\pi]\!]$. For computational reasons, the constraints enforcing that a pipe section must not cross other pipe sections are omitted. They can be checked afterwards, and in case of violation, a manual modification of the pipe design is required.
- For stability reasons, the pipe must be attached to floors or walls of the structure within which it is routed. As a result, there is a set of orientations $\mathcal{O}_c$ allowed within each cell $c \in \mathcal{C}$. Typically, for a rectangular pipe section we impose that the orientation o of each pipe segment passing through a cell c must be orthogonal to a facet of c called a *wall*. If $\overrightarrow{u}$ denotes the normal of this wall, we must ensure that either $\overrightarrow{e_{o,1}}.\overrightarrow{u} = 0$ or $\overrightarrow{e_{o,2}}.\overrightarrow{u} = 0$ (possibility to attach a bracket to one of the sides of the rectangular section).
- Every straight section of the pipe must have a minimum length $L^{min} \in \mathbb{R}^+$. This specification comes from constraints imposed by suppliers.

2.6 Objective Function

The objective in the pipe routing problem is to minimize the overall cost of the pipe, given that each bend b has a cost $C_b \in \mathbb{R}^+$ and each straight section of length ℓ has a cost $\ell\,\mu$ where $\mu \in \mathbb{R}^+$ is a linear cost. The objective is then to minimize the global cost C_π of the pipe defined by:

$$C_\pi = \mu \sum_{i=1}^{N_\pi} \ell_{\pi,i} + \sum_{i=1}^{N_\pi - 1} C_{b_{\pi,i}}$$

3 Routing Plan

In order to route a pipe, we propose to iteratively build its neutral fibre starting from the origin configuration, by making at each step decisions such as the addition of a new bend at the end of the pipe or the crossing of an interface between two adjacent cells. To formalize the approach, the concept of *routing plan* is introduced to represent the decisions made so far on the pipe components.

3.1 Definition

A routing plan s describes, in an abstract way, a neutral fibre composed of N_s successive segments with for each segment i:

- the bend $b_{s,i} \in \mathcal{B}$ applied at the end point of segment i, for $i \in [\![1, N_s - 1]\!]$;
- the sequence of interfaces $\mathcal{I}_{s,i} \subseteq \mathcal{I}$ crossed by segment i, for $i \in [\![1, N_s]\!]$.

Moreover, a routing plan can be terminated or not. When a routing plan is terminated, the neutral fibre has to reach the pipe destination. The set of routing plans is denoted by $\mathcal{S}$. Several data can be derived from the basic definition of a routing plan, including:

- the orientation $o_{s,i} \in \mathcal{O}$ of segment i, for $i \in [\![1, N_s]\!]$; this orientation can be computed from the origin orientation and from the list of bends applied;
- the cell $c_{s,i} \in \mathcal{C}$ to which the end point of segment i belongs, for $i \in [\![1, N_s]\!]$; this cell can be computed from the last interface crossed by segment i.

3.2 Feasibility and Cost

A routing plan $s \in \mathcal{S}$ is *feasible* if it is possible to create a neutral fibre following the choices made in s and satisfying the pipe routing constraints introduced in Sect. 2.5. This feasibility problem can be formulated as a linear program, referred to as LP_s, that contains four kinds of variables:

- *length variables* ℓ_i^s such that, for $i \in [\![1, N_s]\!]$, the real variable ℓ_i^s is the length of the i^{th} segment of the neutral fibre;
- *position variables* $p_i^s = (p_{i,x}^s, p_{i,y}^s, p_{i,z}^s)$ such that, for $i \in [\![1, N_s + 1]\!]$, the real variable $p_{i,x}^s$ (respectively $p_{i,y}^s$ and $p_{i,z}^s$) is the x-coordinate (respectively y-coordinate and z-coordinate) of the i^{th} point of the neutral fibre;
- *interface variables* $q_{i,j}^s = (q_{i,j,x}^s, q_{i,j,y}^s, q_{i,j,z}^s)$ such that, for $i \in [\![1, N_s]\!]$ and $j \in \mathcal{I}_{s,i}$, the real variable $q_{i,j,x}^s$ (respectively $q_{i,j,y}^s$ and $q_{i,j,z}^s$) is the x-coordinate (respectively y-coordinate and z-coordinate) of the intersection between the i^{th} segment of the neutral fibre and an interface j it has to cross;
- *interface distance variables* $\alpha_{i,j}^s$ such that, for $i \in [\![1, N_s]\!]$ and $j \in \mathcal{I}_{s,i}$, the real variable $\alpha_{i,j}^s$ is the distance between the i^{th} point of the neutral fibre and the intersection $q_{i,j}^s$ with the interface j it has to cross.

Linear program LP_s can be formulated as follows:

$$\text{minimize } \mu \sum_{i=1}^{N_s} \ell_i^s + \sum_{i=1}^{N_s-1} C_{b_{s,i}} \tag{1}$$

$$\text{subject to :}$$

$$p_1^s \in \mathcal{P}^{ori.} \tag{2}$$

$$p_{N_s+1}^s \in \mathcal{P}^{dest.} \qquad \text{if } s \text{ is terminated} \tag{3}$$

$$p_{i+1}^s \in \mathcal{P}_{c_{s,i}} \qquad \forall i \in [\![1, N_s]\!] \tag{4}$$

$$\ell_i^s \geq L_{b_{s,i-1}} + L^{min} + L_{b_{s,i}} \qquad \forall i \in [\![1, N_s]\!] \tag{5}$$

$$\overrightarrow{p_i^s p_{i+1}^s} = \ell_i^s \overrightarrow{e_{o_{s,i},3}} \qquad \forall i \in [\![1, N_s]\!] \tag{6}$$

$$q_{i,j}^s \in \mathcal{P}_j \qquad \forall i \in [\![1, N_s]\!] \quad \forall j \in \mathcal{I}_{s,i} \tag{7}$$

$$\alpha_{i,j}^s \leq \ell_i^s \qquad \forall i \in [\![1, N_s]\!] \quad \forall j \in \mathcal{I}_{s,i} \tag{8}$$

$$\overrightarrow{p_i^s q_{i,j}^s} = \alpha_{i,j}^s \overrightarrow{e_{o_{s,i},3}} \qquad \forall i \in [\![1, N_s]\!] \quad \forall j \in \mathcal{I}_{s,i} \tag{9}$$

$$\ell_i^s \in \mathbb{R}^+ \qquad \forall i \in [\![1, N_s]\!] \tag{10}$$

$$p_i^s \in \mathbb{R}^3 \qquad \forall i \in [\![1, N_s + 1]\!] \tag{11}$$

$$q_{i,j}^s \in \mathbb{R}^3 \quad \alpha_{i,j}^s \in \mathbb{R}^+ \qquad \forall i \in [\![1, N_s]\!] \quad \forall j \in \mathcal{I}_{s,i} \tag{12}$$

Constraint 2 states that the neutral fibre must start from the origin cell. Such an inclusion constraint within a convex polyhedron can be expressed as a set of linear constraints. Constraint 3 imposes that if plan s is terminated, the neutral fibre must reach the destination cell. In the same way, Constraints 4 force the successive break points of the neutral fibre to belong to the cell to which they are allocated given plan s. Constraints 5 impose a minimal length on the segments. For the i^{th} segment, this minimum length is obtained from the minimal length L^{min} of straight sections and from the respective contributions

$L_{b_{s,i-1}}$ and $L_{b_{s,i}}$ of the previous and next bends (see Fig. 2). By convention, we assume that $L_{b_{s,0}} = 0$ and $L_{b_{s,N_s}} = 0$, since the first and last segments do not have a previous or a next bend respectively. Constraints 6 define the coordinates of the $(i+1)^{\text{th}}$ point of the neutral fibre from the coordinates of the i^{th} point, the orientation of the i^{th} segment as specified by routing plan s, and the length of this segment. Constraints 7–9 impose that the intersection between the i^{th} segment and an interface j it has to cross must belong both to the interface and to the segment.

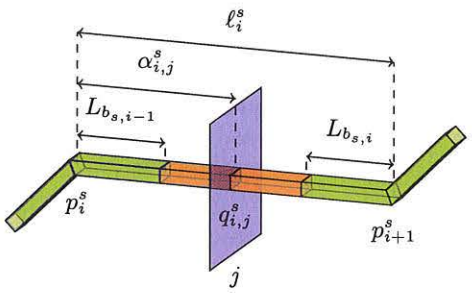

Fig. 2. Portion of a pipe between two successive break points of the neutral fibre (the straight section with variable length is depicted in orange). (Color figure online)

If LP_s does not have a solution, then routing plan s is not feasible and, by convention, its cost is infinite. On the contrary, when there is a solution, routing plan s is *feasible*. In this case, the optimum value $g(s)$ of LP_s is a lower bound on the cost of any pipe that satisfies the constraints defined by routing plan s. Furthermore, the straight sections of the pipe can be rebuilt from the optimal lengths found for the pipe segments, while the bends are already defined by s.

4 Shortest Path Problem Formulation

Based on the routing plan presented above, the pipe routing problem can be seen as a shortest path problem in a graph whose nodes correspond to routing plans and whose arcs represent the basic updates that can be made on routing plans: bend addition, interface crossing, or pipe termination. The goal is then to explore the space of routing plans $\mathcal{S}$ from an empty routing plan starting at the origin cell $c^{ori.}$ with the origin orientation $o^{ori.}$, in order to reach a feasible and terminated plan, ending at destination cell $c^{dest.}$ with a destination orientation in $\mathcal{O}^{dest.}$. Thus, the pipe routing problem can be solved using an A*-like algorithm after formally defining the possible successors of a routing plan in the search space (see Sect. 4.2) and a path-finding heuristic (see Sect. 4.3).

4.1 Search Algorithms

Various heuristic search schemes can be considered to solve shortest path problems. Several of them were compared in [29], showing that hill climbing methods can provide fast results [16,22], while best-first search [1,8,11] and beam search [6,10,30] often offer a better trade-off between solution quality and computation time. Also, compared to beam search, best-first search is more suitable for state spaces in which destination states cannot be reached from some states. We focus here on best-first search techniques, that develop at each step the search node that seems the more promising among all search nodes created so far.

Among the best-first search algorithms, the Weighted A* approach [27] provides competitive results in many fields. Basically, in the A* algorithm [14], the evaluation of a state s is given as $f(s) = g(s) + h(s)$ where $g(s)$ stands for the minimal cost to reach s from the initial state and $h(s)$ stands for the heuristic evaluation of the minimum cost to reach a goal state starting from s. In Weighted A* (denoted WA*), this evaluation is replaced by $f(s) = g(s) + \epsilon \cdot h(s)$ where $\epsilon > 1$ gives a higher weight to the heuristic evaluation, so as to favor the expansion of search nodes corresponding to configurations that seem to be close to the destination. In our case, $g(s)$ is obtained from the value of an optimal solution to LP_s, while three possible versions of $h(s)$ are presented in Sect. 4.3.

4.2 Neighborhood

By definition, a terminated routing plan cannot be expanded. On the contrary, if routing plan $s \in \mathcal{S}$ is not terminated, it can be expanded by adding new routing decisions, as defined below. During plan expansions, only the feasible plans are considered as successors. Indeed, if LP_s is not feasible and if plan $s' \in \mathcal{S}$ extends s, then $LP_{s'}$ is not feasible either because LP_s is a subproblem of $LP_{s'}$.

Bend Addition. Let b be a bend of catalog B. If $N_s < N^{max}$ and if the orientation $r_{M_b}(o_{s,N_s})$ obtained after applying bend b from the current last orientation o_{s,N_s} is acceptable in the current last cell c_{s,N_s} ($r_{M_b}(o_{s,N_s}) \in \mathcal{O}_{c_{s,N_s}}$), then bend b can be added to routing plan s. This adds a new segment to the neutral fibre and definitively allocates the end point of the $N_s{}^{\text{th}}$ segment to cell c_{s,N_s}. Formally, the resulting routing plan s' is defined as follows:

$$N_{s'} = N_s + 1 \tag{13}$$

$$b_{s',i} = b_{s,i} \qquad \forall i \in [\![1, N_{s'} - 2]\!] \tag{14}$$

$$b_{s',N_{s'}-1} = b \tag{15}$$

$$\mathcal{I}_{s',i} = \mathcal{I}_{s,i} \qquad \forall i \in [\![1, N_{s'} - 1]\!] \tag{16}$$

$$\mathcal{I}_{s',N_{s'}} = [\,] \tag{17}$$

Interface Crossing. Let j be a neighbor interface of the current cell, that is $j \in \mathcal{I}(c_{s,N_s})$. Let n_j be the normal to the interface that is oriented towards the

current cell. If the scalar product between the current orientation o_{s,N_s} and n_j is negative and if o_{s,N_s} satisfies the orientation constraints of the cell c_j located on the other side of the interface ($o_{s,N_s} \in \mathcal{O}_{c_j}$), then interface j can be crossed. We also limit ourselves to the destination cells that have not been visited before, meaning that $c_j \neq c_{s,i}$ for $i \in [\![1, N_s]\!]$. Formally, the resulting plan s' is defined as follows:

$$N_{s'} = N_s \tag{18}$$

$$b_{s',i} = b_{s,i} \qquad \mathcal{I}_{s',i} = \mathcal{I}_{s,i} \qquad \forall i \in [\![1, N_{s'} - 1]\!] \tag{19}$$

$$\mathcal{I}_{s',N_{s'}} = \mathcal{I}_{s,N_{s'}} \cup [j] \tag{20}$$

Pipe Termination. Last, if the destination cell and the destination orientation have been reached, meaning that $c_{s,N_s} = c^{dest.}$ and $o_{s,N_s} \in \mathcal{O}^{dest.}$, then routing plan s can be terminated. The resulting plan s' is the same as plan s but it is terminated and it has to reach the destination position (see Constraint 3).

4.3 Trail Heuristic

In order to choose a routing plan s to expand at each step, one key component is the heuristic evaluation of s, that must give an estimation of the minimum cost required to extend s and reach a feasible and terminated routing plan. This section proposes three heuristic evaluations.

Distance as the Crow Flies. A first naive heuristic evaluation is the distance as the crow flies from the end point of the routing plan to the destination position, referred to as $h_{a.c.f.}$. However, this heuristic does not take into account the routing space constraints. The introduced method with this heuristic and $\epsilon = 1$ is used as baseline because of the lack of a usable approach from the literature.

Trail Space. To get more accurate estimations of the distance to the destination, we introduce two other heuristics based on a graph $G(\mathcal{M}, \mathcal{D})$ defining so-called *candidate trails* between some specific points of the routing space. Formally, a *trail* for a routing plan $s \in \mathcal{S}$ is a polyline inside the routing space that connects the end point $p^s_{N_s+1}$ of s as provided by LP_s to the destination polyhedron $\mathcal{P}^{dest.}$. In a trail, the constraints expressing that each orientation change of the polyline must correspond to a bend of the catalog are ignored, the goal being only to estimate the quality of a routing plan. Furthermore, the sequence of interfaces crossed by a trail is called a *channel*.

We propose to discretize the trail space during a preprocessing step by sampling a set of points $\mathcal{M}(i)$ on each interface $i \in \mathcal{I}$ using Bridson's algorithm [7]. It is a maximum Poisson disk sampling method which ensures that sampled points are separated by a *sampling radius* $\rho \in \mathbb{R}^+$, such that when ρ decreases, the density of sampled points increases. Then, for two distinct interfaces i, i' involving a common cell c (i, $i' \in \mathcal{I}(c)$), each point in $\mathcal{M}(i)$ is connected to all the points in $\mathcal{M}(i')$. Last, a set of points $\mathcal{M}\left(\mathcal{P}^{dest.}\right)$ is generated in the destination polyhedron $\mathcal{P}^{dest.}$. The points in $\mathcal{M}\left(\mathcal{P}^{dest.}\right)$ are connected to points on the interfaces of $c^{dest.}$. The graph obtained is denoted $G(\mathcal{M}, \mathcal{D})$.

Trail Length Heuristic. During a preprocessing step, the distance L_M from each point $M \in \mathcal{M}$ to the destination cell as well as the corresponding shortest trail is computed using a Dijkstra procedure. Then, the remaining cost to the destination for a routing plan $s \in \mathcal{S}$ can be evaluated based on the length of the shortest trails in $G(\mathcal{M}, \mathcal{D})$ (see Fig. 3). The estimation is further improved by using the shortest trail for an *extended routing plan* $\tilde{s} \in \mathcal{S}$ that can be computed by crossing as many interfaces as possible in the channel of the shortest trail from plan s without adding any bends. The feasibility of the extended plans is evaluated using the linear programs, whose solutions can be reused when the extended plans are expanded in turn. Finally, this heuristic is defined by:

$$h_{length}(s) = \begin{cases} \mu \cdot \min_{M \in \mathcal{M}(\mathcal{P}^{dest.})} \|\overrightarrow{p^{\tilde{s}}_{N_{\tilde{s}}+1} M}\| & \text{if } c_{\tilde{s},N_{\tilde{s}}} = c^{dest.}, \\ \mu \cdot \min_{M \in \cup_{i \in \mathcal{I}\left(c_{\tilde{s},N_{\tilde{s}}}\right)} \mathcal{M}(i)} \left(\|\overrightarrow{p^{\tilde{s}}_{N_{\tilde{s}}+1} M}\| + L_M \right) & \text{otherwise.} \end{cases} \tag{21}$$

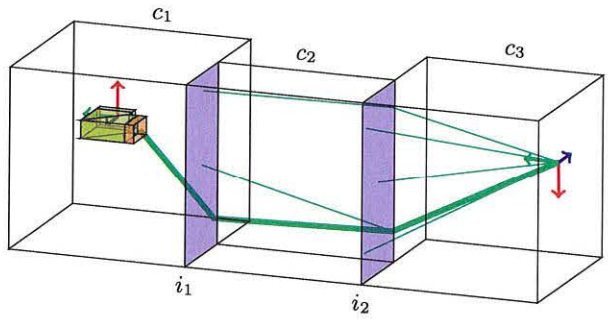

Fig. 3. Trail heuristic on a simple routing case (trails are depicted in green). (Color figure online)

Trail Cost Heuristic. Previous heuristics only estimate the lineic contribution to the cost of a pipe. However, the polyline $[M_1, ..., M_K]$ of the shortest trail for extended plan $\tilde{s}$ can also be used to estimate the cost of the remaining bends that will be added to the pipe to reach the destination. This approach is described in Algorithm 1, that returns a heuristic evaluation referred to as h_{cost}. This algorithm analyzes the successive orientation changes on the trail and tries to reproduce these changes with the bends of the catalog. It estimates the bend cost using a function σ that takes as an input the orientation o of the pipe and a vector $\overrightarrow{u}$, and that returns as an output a quantity $\sigma(o, \overrightarrow{u}) = \overrightarrow{e_{o,3}} \cdot \frac{\overrightarrow{u}}{\|\overrightarrow{u}\|}$ estimating through a scalar product the angular proximity between the main direction $\overrightarrow{e_{o,3}}$ of orientation o and the direction defined by $\overrightarrow{u}$. The closer $\sigma(o, \overrightarrow{u})$ is to 1, the closer o is from direction $\overrightarrow{u}$. Note that with such an approach, we ignore the orientation of the pipe section around the neutral fibre. Last, from the last orientation o reached when following the trail, Algorithm 1 also adds the

cost $Dijkstra\left(o, \mathcal{O}^{dest.}\right)$ of the best bend combination that reaches a destination orientation from o. Such a cost can be precomputed for any reachable orientation o, before using Algorithm 1.

Algorithm 1. Evaluate $h_{cost}(s)$

Require: $s \in \mathcal{S}$, shortest trail polyline $[M_1, ..., M_K]$ from extended plan $\tilde{s}$.

```
o ← o_{s,N_s}
C_bends ← 0
for k ∈ [[1, K − 1]] do
  repeat
    improvement ← false
    b* ← argmax_{b∈B} ( σ ( r_{M_b}(o), M_k M_{k+1}→ ) )
    if σ ( r_{M_b*}(o), M_k M_{k+1}→ ) > σ ( o, M_k M_{k+1}→ ) then
      improvement ← true
      C_bends ← C_bends + C_b*
      o ← r_{M_b*}(o)
    end if
  until improvement = false
end for
return h_length(s) + C_bends + Dijkstra(o, O^{dest.})
```

Admissibility. It is important to note that heuristic functions h_{length} and h_{cost} are not necessarily admissible, as they can overestimate the real cost required to reach the goal configuration. Additionally, even with the distance as the crow flies, the state evaluations are not monotonic, meaning that for a state s' extending s, we might have $f(s') < f(s)$ (proof omitted for space reasons, but basically it is possible to define an example in which the intermediate end point implicitly chosen when solving LP_s does not belong to an optimal path). These remarks imply that the first feasible and terminated routing plan reached by A*-like algorithms with these heuristics is not necessarily optimal in our case.

5 Experiments

The shortest path formulation presented in Sect. 4 for the pipe routing problem can be solved using a Weighted A* (WA*) search hybridized with a linear program which evaluates the cost of routing plans. This hybrid approach has been implemented using the Java language and the simplex solver from `Apache Commons Math 3.6.1` (but other LP solvers might be faster). It has been tested on four routing problems of increasing complexity, with the three proposed heuristics $h_{a.c.f.}$, h_{length} and h_{cost}. Experiments were conducted on an Intel 2.70 GHz processor with 16 GB of RAM. In absence of a reusable approach from the literature, the baseline is the proposed method with $h_{a.c.f.}$ and $\epsilon = 1$.

5.1 Test Cases

In practice, the proposed method is used by adding conflicting obstacles iteratively into the cell decomposition. Thus, it is possible to reduce industrial cases to simple setups like the one presented on Fig. 4. The dimensions of this routing space are 380 by 120 by 120. Instances 1, 2, 3 and 4 split this routing space into respectively 8, 34, 62 and 78 cells after adding respectively 0, 18, 42 and 58 obstacles into the cell decomposition. All instances aim at connecting configuration θ_s to configuration θ_d using a maximum number of 100 bends (that defines the maximum number of segments). The bend catalog contains 90° and 45° bends that have a common cost $C_b = 100$. The linear cost is set to 1. In the industry, such instances have to be solved many times, so a solution must be found quickly. For this purpose, the runtime has been limited to 3 min here, and the search is stopped when the first solution is found.

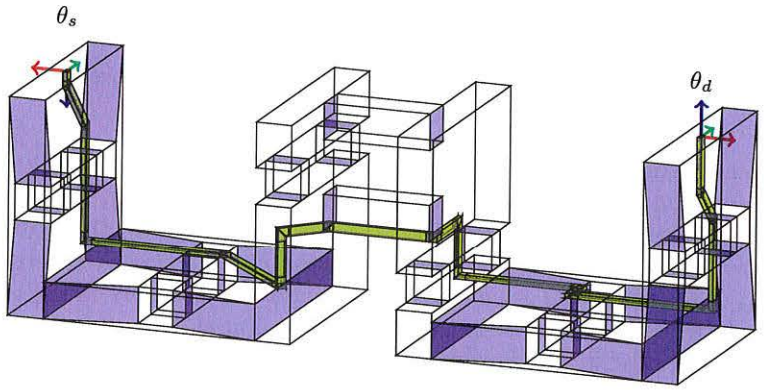

Fig. 4. Configurations of the instances, with the routing space and the solution of instance 2 found using WA* search and heuristic h_{cost} with $\epsilon = 1.2$ and $\rho = 5$.

5.2 Results and Discussion

Our approach has been tested with the WA* search using ϵ values set to 1 (corresponding to A*), 1.2, 1.5, 2, 3 and 5. For the trail heuristics, the sampling radius ρ is set to 5. Figure 5 shows the performances and the number of visited plans for the different heuristics. For clarity reasons, only the best values of ϵ are shown for each heuristic.

As expected, heuristic $h_{a.c.f.}$ which uses the distance as the crow flies does not succeed to solve difficult instances and visits many more routing plans than other heuristics. In the same way, the trail length heuristic h_{length} requires a high value of $\epsilon = 5$ to be able to solve more complex instances, and it does not manage to solve Instance 4 within an acceptable runtime. Nevertheless, when heuristics $h_{a.c.f.}$ and h_{length} succeed, they provide good solutions.

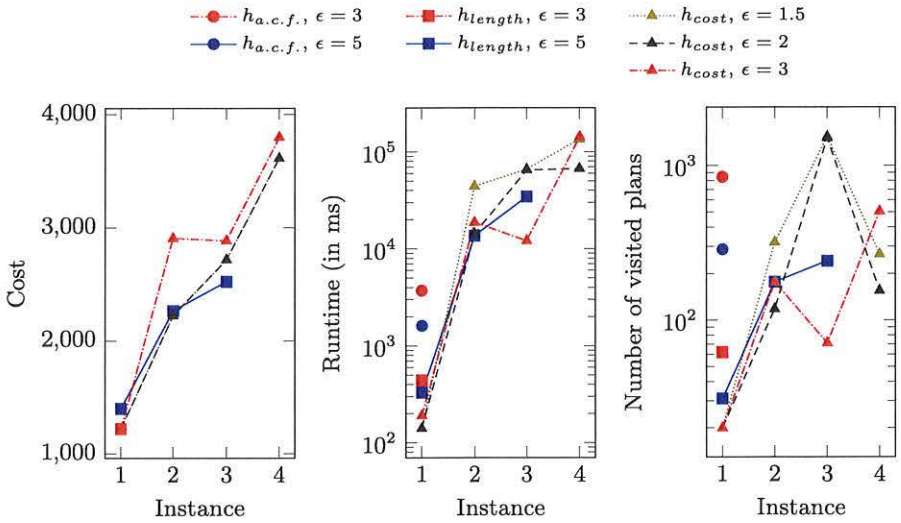

Fig. 5. Results of the three heuristics with $\rho = 5$ on the four test case instances.

In comparison, heuristic h_{cost} outperforms the previous ones when the number of cells increases, that is when there are more obstacles to avoid. Indeed, it solves all instances within reasonable runtimes, even with smaller values of ϵ, and the solutions obtained on the smallest instances are competitive with $h_{a.c.f.}$ and h_{length}. Difficult cases are solved after about one minute and the pipes found are acceptable by a human designer, as shown on Fig. 4.

In a second experiment, heuristic h_{cost} has been tested using ρ values of 1, 5, 10, 25 and 50 to evaluate the impact of the sampling radius on the performances. The results are shown on Fig. 6. WA*-search tends to visit more routing plans when the sampling radius ρ decreases, meaning that more points are sampled on each interface. Consequently, the resolution of the routing instances takes more time. However, increasing the sampling density seems to improve the cost of the solutions. This trend must be confirmed on more test cases, but the adjustment of the sampling radius would be a trade-off between the runtime and the quality of the solutions. On our instances, the best complete tuning uses $\epsilon = 2$ and $\rho = 5$, which provides the solution given in Fig. 4.

Obviously, the regularity of the solutions depends on the bend cost. The instances used in this paper strongly penalize the bends in comparison with the linear cost, which favors the regularity of the pipes. Bend costs closer to the lineic cost would lead to less regular optimal pipes and the approach would provide solutions that might be rejected by designers because of a high number of bends.

Last, real test cases have confirmed that linear program LP_s does not prevent a pipe from colliding with itself. It particularly happens to correct an infinitesimal gap on one axis between the origin and destination positions. This case can be fixed using a tolerance on positions, but the phenomenon also occurs when

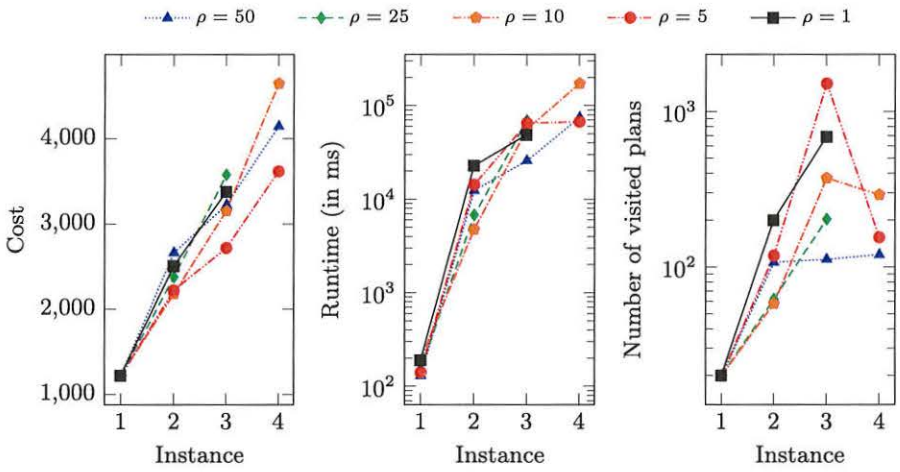

Fig. 6. Impact of the sampling radius ρ using WA*-search with $\epsilon = 2$ and h_{cost}.

the pipe orientation has to change in a small space. The automated management of self interfering constraints is a future work direction.

6 Conclusion

This paper presented an algorithm for routing a single pipe in a continuous 3D routing space divided into convex cells, using both orthogonal and non-orthogonal bends. The approach is able to take into account the orientation of the pipe section, which is particularly useful for rectangular pipes. It is based on a combination between weighted A* search in the discrete space of routing plans and linear programming for evaluating the feasibility and the cost of a routing plan in the continuous routing space. Several heuristics were proposed and compared on realistic instances. The best one, based on a quick estimation of the remaining trail to reach the destination, provides acceptable solutions within reasonable runtimes and can be used on industrial cases.

For future work, the performance of the proposed approach could be improved by adding to the routing plan the definition of the next interface that should be crossed. The underlying idea is that the linear program could exploit such a specification to generate a better intermediate end point. Furthermore, the linear program LP_s is solved from scratch at each iteration, but it could be warm-started using the solution formed for the prior routing plan. Additionally, instead of A* or weighted A*, a full Mixed Integer Linear Programming (MILP) model could be tested, even if experiments performed on obstacle-free routing spaces showed that getting results within a few seconds can be challenging for MILP. Last, the method must be extended to the multiple pipe routing problem using an high level approach like in [5].

References

1. Aine, S., Chakrabarti, P., Kumar, R.: AWA*-a window constrained anytime heuristic search algorithm. In: IJCAI, pp. 2250–2255 (2007)
2. Ando, Y., Kimura, H.: An automatic piping algorithm including elbows and bends. J. Jpn. Soc. Naval Architects Ocean Engineers **15**, 219–226 (2012)
3. Asmara, A., Nienhuis, U.: Automatic piping system in ship. In: International Conference on Computer and IT Application (COMPIT). Citeseer (2006)
4. Belov, G., Czauderna, T., Dzaferovic, A., Garcia de la Banda, M., Wybrow, M., Wallace, M.: An optimization model for 3d pipe routing with flexibility constraints. In: Beck, J.C. (ed.) CP 2017. LNCS, vol. 10416, pp. 321–337. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66158-2_21
5. Belov, G., Du, W., Garcia de al Banda, M., Harabor, D., Koenig, S., Wei, X.: From multi-agent pathfinding to 3D pipe routing. In: Symposium on Combinatorial Search (SoCS) (2020)
6. Bisiani, R.: Beam search. Encycl. Artif. Intell. **2**, 1467–1468 (1992)
7. Bridson, R.: Fast Poisson disk sampling in arbitrary dimensions. SIGGRAPH Sketches **10**, 1 (2007)
8. Ebendt, R., Drechsler, R.: Weighted A* search-unifying view and application. Artif. Intell. **173**(14), 1310–1342 (2009)
9. Fan, X., Lin, Y., Ji, Z.: The ant colony optimization for ship pipe route design in 3D space. In: 2006 6th World Congress on Intelligent Control and Automation, vol. 1, pp. 3103–3108. IEEE (2006)
10. Furcy, D., Koenig, S.: Limited discrepancy beam search. In: IJCAI (2005)
11. Furcy, D., Koenig, S.: Scaling up WA* with commitment and diversity. In: IJCAI, pp. 1521–1522 (2005)
12. Furuholmen, M., Glette, K., Hovin, M., Torresen, J.: Evolutionary approaches to the three-dimensional multi-pipe routing problem: a comparative study using direct encodings. In: Cowling, P., Merz, P. (eds.) EvoCOP 2010. LNCS, vol. 6022, pp. 71–82. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12139-5_7
13. Guirardello, R., Swaney, R.E.: Optimization of process plant layout with pipe routing. Comput. Chem. Eng. **30**(1), 99–114 (2005)
14. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
15. Hightower, D.W.: A solution to line-routing problems on the continuous plane. In: Proceedings of the 6th Annual Design Automation Conference, pp. 1–24 (1969)
16. Hoffmann, J.: Extending FF to numerical state variables. In: ECAI pp. 571–575. Citeseer (2002)
17. Ikehira, S., Kimura, H., Ikezaki, E., Kajiwara, H.: Automatic design for pipe arrangement using multi-objective genetic algorithms. J. Jpn Soc. Naval Architects Ocean Engineers **2**, 155–160 (2005)
18. Ito, T.: A genetic algorithm approach to piping route path planning. J. Intell. Manuf **10**(1), 103–114 (1999)
19. Jiang, W.Y., Lin, Y., Chen, M., Yu, Y.Y.: A co-evolutionary improved multi-ant colony optimization for ship multiple and branch pipe route design. Ocean Eng. **102**, 63–70 (2015)
20. Kim, S.H., Ruy, W.S., Jang, B.S.: The development of a practical pipe auto-routing system in a shipbuilding CAD environment using network optimization. Int. J. Naval Architecture Ocean Eng. **5**(3), 468–477 (2013)

21. Kimura, H.: Automatic designing system for piping and instruments arrangement including branches of pipes. In: International Conference on Computer Applications in Shipbuilding (ICCAS), pp. 93–99 (2011)
22. Koenig, S., Sun, X.: Comparing real-time and incremental heuristic search for real-time situated agents. Auton. Agents Multi-Agent Syst. **18**(3), 313–341 (2009)
23. Lee, C.Y.: An algorithm for path connections and its applications. IRE Trans. Electron. Comput. **3**, 346–365 (1961)
24. Liu, L., Liu, Q.: Multi-objective routing of multi-terminal rectilinear pipe in 3D space by MOEA/D and RSMT. In: 2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM), pp. 462–467. IEEE (2018)
25. Medjdoub, B., Bi, G.: Parametric-based distribution duct routing generation using constraint-based design approach. Autom. Constr. **90**, 104–116 (2018)
26. Park, J.H., Storch, R.L.: Pipe-routing algorithm development: case study of a ship engine room design. Exp. Syst. Appl. **23**(3), 299–309 (2002)
27. Pohl, I.: First results on the effect of error in heuristic search. Mach. Intell. **5**, 219–236 (1970)
28. Sakti, A., Zeidner, L., Hadzic, T., Rock, B.S., Quartarone, G.: Constraint programming approach for spatial packaging problem. In: Quimper, C.-G. (ed.) CPAIOR 2016. LNCS, vol. 9676, pp. 319–328. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33954-2_23
29. Wilt, C.M., Thayer, J.T., Ruml, W.: A comparison of greedy search algorithms. In: 3rd Annual Symposium on Combinatorial Search (2010)
30. Zhou, R., Hansen, E.A.: Beam-stack search: integrating backtracking with beam search. In: ICAPS, pp. 90–98 (2005)
31. Zhu, D., Latombe, J.C.: Pipe routing-path planning (with many constraints). In: Proceedings. 1991 IEEE International Conference on Robotics and Automation, pp. 1940–1941. IEEE Computer Society (1991)