Activity Planning for Assistive Robots using Chance Constrained Stochastic Programming

Bevilacqua, P., *Member, IEEE,* Frego, M., Palopoli, L., *Member, IEEE* Fontanelli, D. *Senior Member, IEEE*

***Abstract*—In this paper, we present a framework for planning an activity to be executed with the support of a robotic navigation assistant. The two main components are the Activity and the Motion Planner. The Activity Planner composes a sequence of abstract activities, chosen from a given set, to synthesise a plan. Each activity is associated with a point of interest in the environment and with probabilistic parameters that depend on the plan, which are characterised by simulations in realistic scenarios. The low-level action to pass from an activity to the next is handled by the Motion Planner, which secures the physical feasibility of the chosen actions and their compatibility with the constraints posed by the user and the environment. Indeed, the final plan must respect the user constraints and optimise his/her satisfaction from the activity. We show a possible model for the problem as a chance constrained optimisation along with an efficient technique to find high quality solutions.**

***Index Terms*—Assistive Robots, Activity Planning, Motion Planning, Chance Constrained Opt., Integer Programming.**

I. Introduction

The possible ways robotic technologies can be used to assist humans are virtually beyond count. As a few examples, robotic prosthetics like hands [1] or legs can be used to replace missing or severely injured limbs [2]. Robots can also be used for post-injury rehabilitation [3] or as home assistants [4]. In this paper, we are interested in a particular category: navigation and activity assistants. The objective is, in this case, to spur senior users into frequent and compelling social activities beyond the walls of their homes by offering physical and cognitive support. A robot of this kind can be disguised as an ordinary walker, but under the hood it contains a package of technologies to sense the surrounding environment, to suggest a sequence of actions and a safe path, to guide the user along this path and to react to unexpected situations in the environment. Robots of this kind have been proposed in the context of several research projects [5], [6], [7].

Assistive Robotics. A key component of robotic assistants is the planner, providing the users with sets of possible activities to perform (e.g. the visit to a shopping mall, a museum, etc.), based on their interests and requirements, and giving them assistance and support during the execution of the chosen activities, to guarantee safety, comfort and satisfaction. At runtime, a monitoring component controls in what measure the user follows the plan, and, if necessary, triggers an update to tackle relevant deviations. The development of a planner has several facets: the plan generated by the robot has indeed to comply with the complex and diversified physical and psychological requirements associated with the user. In this context, there is not a viable one-size-fit-all solution: each user is an individual for whom different barriers have to be lowered and different motivations have to be triggered. Moreover, the high level activity plan has to be translated to a sequence of primitive and executable actions, i.e. elementary paths, that can be followed by the user and by its robot assistant. The software component responsible for the execution of the paths is the Motion Planner that, at this "lower" level, needs to consider also the robot kinematics and dynamics, and the presence of additional constraints coming from the environment (a wall, a temporary obstruction due to maintenance operations, etc.).

P. Bevilacqua, M. Frego and L. Palopoli are with the Dep. of Information Engineering and Computer Science; D. Fontanelli is with the Dep. of Industrial Engineering, University of Trento, Italy.

Related work (High-level Planning). The high-level planning is abstracted as a graph, where the nodes represent relevant points of interest and the edges are weighted by stochastic values such as distance and travel time. In literature, this kind of optimisation problems, focusing on the search of the optimal order to visit a given set of destinations, is known as the Multi-Goal Motion Planning Problem. A possible solution strategy for these problems is based on the reduction to Travelling Salesman Problem (TSP) instances, and on the adoption of one of the large number of effective TSP algorithms and solvers available. This kind of approach is found for example in [8], where the TSP is solved with a common greedy technique, based on the construction of a Minimum Spanning Tree, while the tree arcs are sequentially refined and made feasible by the lower-level motion planner. Various applications in literature require a strong interplay between planners working at different levels of abstraction. For example, in [9], the Authors present a planner for both terrestrial and flying agents, accounting also for the dynamics of the different vehicles, by means of a multi-layered approach.

Other kinds of problems bearing some similarities with the considered Activity Planning are the Partial Satisfaction Planning [10], and the variant of the Orienteering Problem (OP) [11] known as the Tourist Trip Design Problem (TTDP). In TTDP, given a set of attractions to visit, each with a certain score given according to the user preferences, the solver has to choose the attractions to include in the plan, and the optimal order of the visits, given some constraints on the maximal duration of the plan. Generally, most of the solutions available in literature adopt metaheuristic optimisation algorithms [12]. A survey illustrating the TDDP and its variants, and discussing possible solutions, is available in [13].

However, to the best of the Authors' knowledge, none of the existing solutions deals with the kind of problems that we are called to solve for the Activity Planner, characterised by the presence of different constraints expressed in probabilistic terms. For example, the objective of the TSP is to determine the shortest path visiting all the vertices of the graph. On the contrary, we can select a subset of nodes to visit, in order to maximise the user satisfaction, according to his/her preferences, respecting both deterministic and stochastic constraints. On the other hand, in general, all the existing solutions to the TTDP focus on deterministic constraints only, or on the modelling of a single stochastic constraint to be satisfied. Therefore, these metaheuristic approaches handle only one nondeterministic constraint, whereas the problem tackled in this paper requires the effective handling of a number of them.

Related work (Low-level Planning). A strong body of scientific results has addressed the problem of motion planning. A review of these methods can be found in a rather complete textbook [14]. We can attempt a rough classification among algorithms based on cell decomposition [15], [16], on sampling [17] and on potential fields [18].

Contribution of the paper. The focus of this paper is the design of an integrated planning tool, that can be employed for the generation of sequences of tasks and activities to be performed by older adults in public spaces with the support of assistive robotic rollators. To this end, our approach demands a strong interaction between the motion planner, working at a lower level of abstraction, and the activity planner, working at a higher level, with the intended goal to produce a sequence of smooth, feasible trajectories, visiting a subset of the available points of interest (POIs). The activity planning problem poses a set of challenges, that cannot be solved reasoning solely at a high level or at a low level of abstraction. On one hand, at an higher level it is simpler to model and capture "complex" elements such as user requirements and preferences. However, a graph based representation of the environment may not be adequate to model operations in unstructured and "open" environments, such as public spaces. Also, standard motion planning algorithms are not suitable to handle all the complex requirements that have to be considered for the synthesis of activity plans (e.g. visits to sequences of points of interest). Therefore, a strict convergence between low and high level planning is necessary for an effective solution of the problem. It is worthwhile to note that even with this decomposition of the Activity Planning, that is modelled at two different levels of abstraction, the high level optimisation problem belongs to the NP-hard class. Hence, it cannot be solved with standard graph search algorithms, e.g. A*, but it requires the adoption of more powerful optimisation tools.

The environment where an activity takes place contains a set of POIs, each with a certain score, depending on the user preferences. In addition, users may have different constraints and requirements, with different levels of criticality (i.e. different acceptable probabilities of violation). The Activity Planner uses the information on the environment and the user profile, to generate plans, i.e. sequences of tasks. Each of these tasks corresponds to a motion between a pair of POIs, and it is associated with physical parameters such as length, travel time, etc., modelled probabilistically. The characterisation of these parameters is performed offline by running the motion planner over a large number of different scenarios. Constraints

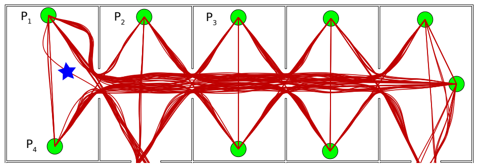


Fig. 1. Examples of possible trajectories among pairs of nodes. The blue star denotes a deviation to bypass a temporary obstacle.

expressed in probabilistic terms can be formalised by recasting the problem as a Chance Constrained Stochastic Programming (CCSP) with integer decision variables. Thus, in order to effectively deal with probabilistic constraints, we model Activity Planning problems as CCSP instances. Considering the limited size of the planning instances to be solved, it is possible to apply exact algorithms, yielding the optimal solution within reasonable and acceptable computational times. In addition, we propose also a hierarchical decomposition of full problems, applicable to larger examples of problems. As shown in the paper, this expedient enables us to produce very high quality sub-optimal solution in an acceptable time (a few minutes) even for unrealistically large scale problems.

The generated activity is then refined by the Motion Planner into an executable plan. This last aspect is briefly summarised in this paper, being the main focus on the Activity Planner.

Organisation of the paper. In Section II, we will offer a detailed description of the problem and present our approach. In Sections III and IV, we will show how to formalise and solve the activity planning problem. In Section V, we will discuss the Motion Planner technology that we combine with the Activity Planner in order to characterise the actions and to refine the Activity Plan. In Section VI, we show experimental results and validation, finally, in Section VII, we report our conclusions and future work.

II. The Activity Planning Problem

The Problem. The goal of the Activity Planner is to produce plans for activities designed for an old person, henceforth called *user*, with the help of an assistive robot. An activity is conducted in wide public area such as a museum or a shopping mall (called *environment*), where a number of points of interest are located. A mild hypothesis is that both the map layout and the position of the POIs are known by the planner, as an example, a museum could provide a layout of the building labelled with the collection of exhibited items (the POIs), together with a statistic of the usual visit times.

The POIs are tagged with different categories (e.g. fossils, mammals, etc) in order to meet the interests of the visitor, who can assign a different rating to each of them and ask the Activity Planner to organise the optimal visit tailored to her/his specific needs: the *activity plan*. It is thus feasible and computationally reasonable to consider a number of POIs of about one hundred. For instance, in Figure 1 the green dots represent the POIs and the red paths possible connections. The goal of the Activity Planner is the choice of a selection of these POIs to design an activity. As an example, an optimal sequence may be given by P_1, P_4, P_2 and P_3.

To evaluate the score of an activity plan, a set of tags is assigned to each POI. Based on the score given by the user to each category, and stored in her/his user profile, the planner can determine the value of each POI as the sum of the values given to each of its tags. Users scores can be collected with a standard evaluation scheme, for instance based on a simple star rating, or votes from 0 to 10, or with a percentage of interest. Another possibility is to have the score imported from a social network, e.g., it could be the average evaluation of the POI reported by the participants with a profile similar to the user's. In addition, it is possible to add goals in terms of metrics related to the physical condition of the user, such as distance walked or calories burnt. We address next the principal constraints that can be given to the Activity Planner.
1. **Environment constraints**: the geometric information on the environment is stored in a specialised database (SpatiaLite [19]) as a set of polygons, along with the position of the POIs and their properties. Because the environment has specific geometries and configurations, the robotic walker can move only in certain areas (passageways). Fixed obstacles impose constraints on the synthesis of the trajectory. Moreover, some environmental constraints change with time, e.g., a passageway may be temporarily obstructed by a crowd or a room may be closed for maintenance. Thus, according to these events, pairs of POIs may or may not be directly connected.
2. **Physical constraints**: we can model the ensemble user–walker as a car-like vehicle with rear wheel drive, as the user pushes the assistive robot, which is actuated on the front wheels. If the front wheels are free and the actuation is on the rear, the resulting model is an unicycle-like vehicle [20]. In both cases we need to account for nonholonomic constraints.
3. **User constraints**: the user's physical or psychological conditions are encoded in the so-called user constraints. Examples of user constraints can be upper bounds on the duration of the walking activity before taking a break or, on the contrary, the enforcing of a minimum distance to be walked. Another case can be the availability of a seat or of a toilette in the vicinity. Those constraints may have different levels of criticality: for some users it could be compulsory to keep clear of busy areas, but only desirable to have a certain number of stops during the walk. Different criticality levels are modelled assigning different probabilities of satisfaction to the constraints (mandatory constraints are satisfied with probability 1). As sketched in Figure 2, user requirements can be specified and inserted in the user profile also by a doctor, a relative, a caregiver, and, of course, directly by the user.
The Approach. The adopted method splits the problem between Motion Planning and Activity Planning. The former involves the physical motion of the robot between two configurations, satisfying the kinematic and dynamic constraints, while the latter optimises the choice and the sequence of high level actions that build the plan. Visits to a shopping mall or a museum are proposed to the user by a recommendation system that accounts for her/his preferences, or by an external agent, such as a relative, a doctor, friends, social media, etc.

We propose a solution in two phases: an offline statistical preprocessing of the known environment (map and POIs); an online construction of the plan that accounts for all the above discussed constraints and that is given to the Motion Planner in order to be executed. The interaction between the Activity and the Motion Planner is shown in Figure 2. An elemental unit of the plan is an action $\mathcal{Y}_{i,j}$, that corresponds to a motion from POI number i (denoted as POI$_i$) to POI number j (POI$_j$). The duration of the action is denoted by $\mathcal{T}\left(\mathcal{Y}_{i,j}\right)$, and depends both on the walking speed of the user and on the length of the path, which, in turn, depends on the general conditions of the environment. Therefore, we model $\mathcal{T}\left(\mathcal{Y}_{i,j}\right)$ as a random variable, and estimate its probability distribution via simulations. The environment is represented geometrically by specifying the location of the POIs and the layout of the map and of the static obstacles. As an example, consider the map of a floor of a shopping mall annotated with walls and pillars. This base model is used in the simulations as a static map, randomly populated with moving elements, like groups of people walking, and temporarily closed areas. Figure 1 shows an example of such simulations: the blue star identifies a particular trajectory deviating from the shortest one, to avoid a temporary obstacle. In each simulation, different dynamic elements are created on the basis of probability distributions, estimated according to observations of the real environment. Specifically, different scenarios are characterised by the density of the dynamic elements (e.g. people), and by the number and extension of unaccessible areas. Furthermore, different kinds of users, with varying health conditions, are simulated. For example, we consider both "healthy" users, with a walking speed exceeding 1m/s, and debilitated users, moving at 0.5m/s or less. For each simulation, the optimal trajectory between POI$_i$ and POI$_j$ avoiding both static and dynamic obstacles is found. The objective function depends both on the length of the trajectory and on its shape (e.g., frequent curvature changes are perceived as uncomfortable [20]). The analysis of the data extracted from the simulations yields the probability distribution of the duration $\mathcal{T}\left(\mathcal{Y}_{i,j}\right)$ of an action for each user profile in each considered scenario. Similarly, the probability distributions of the travelled distance $\mathcal{D}\left(\mathcal{Y}_{i,j}\right)$ and of the burnt calories $\mathcal{C}\left(\mathcal{Y}_{i,j}\right)$ are obtained. This information is stored in the Knowledge Base (see Figure 2 for reference). A key role for planning decisions is played by the failure rate of $\mathcal{Y}_{i,j}$. Whenever in a simulation the Motion Planner fails to produce a feasible path, a counter is incremented. When the simulation is completed, it is used to compute the failure rate. This pre-processing is done offline, indeed, these statistics can be periodically updated with the new collected data.

At run-time, all the relevant information is retrieved, and the Activity Planner returns a plan which is compliant with the domain and the user profile. To this end, the current

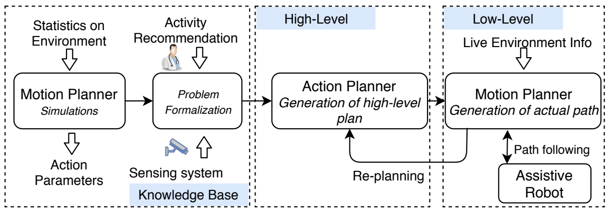


Fig. 2. Architecture of the Activity Planner framework.

operating scenario is estimated by the sensors deployed in the environment, and the probability distributions are retrieved accordingly. Then, the planning problem can be formulated in a suitable, standard format, accepted by the off-the-shelf optimisation tool chosen, as detailed in the next section. The result of this optimisation is a plan composed of a list of motion tasks, given as input to the Motion Planner. Thus, in most of the cases, the Motion Planner produces a path fulfilling the elder's desires and needs, synthesised according to their level of criticality. It is, however, possible that some unique setting occurs, where no valid plans can be produced. In these circumstances, an exception is raised and a new run of the Activity Planner is triggered, accounting for the current unforeseen situation, to adjust the original schedule. Nodes and edges corresponding to unavailable POIs and actions are removed from the graph, and the remaining resources (in terms of time, distance and satisfaction of the various constraints) updated. The new problem has the same formulation of the original one, but a smaller size. An example of such a re-planning is presented in Section VI.

The threshold on the failure rate is a useful tuning knob. If we set it to 0, it means that we are considering a worst case approach: we consider an action as eligible only if the Motion Planner did not fail for any simulation.

III. Formalisation of the Activity Planning

In Section II we described the abstraction of the map as a graph with nodes (the POIs) and edges (the actions): an action $\mathcal{Y}_{i,j}$ connects POI$_i$ to POI$_j$. As explained above, the operating conditions of the current scenario are determined by the remote sensors deployed in the environment. With the Boolean vector variable $\boldsymbol{p}$ we represent the POIs and we set $p_i = 1$ if POI$_i$ is visited, and $p_i = 0$ otherwise. For the actions, similarly, we use a Boolean variable $Y_{i,j}$ that is equal to 1 when the corresponding action $\mathcal{Y}_{i,j}$ is part of the solution. These binary variables naturally model the visit of a node or the traversal of an edge. Thus, the problem is recasted to an Integer Programming formulation. The objective function to be optimised is related to the sum of the scores of the visited POIs. Letting c_i be the score associated with POI$_i$ for the current user, the objective function can be written as $\sum_i c_i p_i = \boldsymbol{c}^T \boldsymbol{p}$, where $\boldsymbol{c}$ is the vector of scores. This function can be extended with features related to the physical conditions of the user, e.g. a term $\sum_{i,j} Y_{i,j} \mathbb{E}\left(\mathcal{D}\left(\mathcal{Y}_{i,j}\right)\right)$ accounting for the distance walked, or a term $\sum_{i,j} Y_{i,j} \mathbb{E}\left(\mathcal{C}\left(\mathcal{Y}_{i,j}\right)\right)$ for the calories spent. Being these values expressed in probability, we consider their expected value $\mathbb{E}\left(\cdot\right)$.

The constraints are defined on the basis of the user profile, and can be classified as hard or soft constraints. Hard constraints are enforced by linear inequalities modelling specific user needs and requirements, for instance maximum path length, minimum calories spent, and so on. These constraints involve a subset of nodes and edges, through the corresponding Boolean variables $\boldsymbol{p}$ and $Y_{i,j}$ for certain indexes i, j. As an example, suppose that the user cannot walk for more than 15 minutes, the resulting constraint takes the form $Y_{i,j} = 1 \implies \mathcal{T}(\mathcal{Y}_{i,j}) \leq 15$. This can be reformulated as $Y_{i,j} \cdot \mathcal{T}(\mathcal{Y}_{i,j}) \leq 15$. Soft, non-critical constraints, to be satisfied only with a certain probability, are instead modelled as $Y_{i,j} = 1 \implies \mathbb{P}\left(\mathcal{T}(\mathcal{Y}_{i,j}) \leq 15\right) \geq 1 - \alpha$ (or, equivalently, $\mathbb{P}\left(Y_{i,j} \cdot \mathcal{T}(\mathcal{Y}_{i,j}) \leq 15\right) \geq 1 - \alpha$), where α quantifies the probability of violation.

Additional constraints are required to guarantee the sequential structure and connectivity of the visit, and to prevent sub-tours, which are cycles disconnected from the rest of the solution path. We can effectively avoid sub-tours by applying *lazy constraints* [21], which are widely employed for the solution of the Travelling Salesman Problem. These constraints are natively supported by most state-of-the-art numerical solvers, such as Gurobi, Cplex and GLPK. Indeed, the exponential number of possible sub-tours (i.e. cycles) in a graph, requires a dynamic generation of the related constraints, during the search for the optimal solution. When a new solution is generated, it is checked for possible sub-tours. In the presence of one or more of them, new lazy constraints are added. Each sub-tour elimination constraint is formulated as $\sum_{i,j} Y_{i,j} \leq \dim - 1$, where the sum is computed over the indices (i, j) of its edges, while $\dim$ is the size of the cycle.

It is also possible to solve the problem without the lazy constraints, but at the price of adding other variables and constraints: by introducing integer variables $\boldsymbol{t}$ having the same size of $\boldsymbol{p}$, we can model the order of the visits to the POIs. With these variables, it is possible to avoid loops in the solution by imposing the condition $Y_{i,j} = 1 \implies t_j = t_i + 1$, or, in algebraic form, $(Y_{i,j} - 1)m + 1 \leq t_j - t_i \leq (1 - Y_{i,j})m + 1$, with m a large, positive constant. Whenever $Y_{i,j} = 1$, then $1 \leq t_j - t_i \leq 1$, yielding $t_j = t_i + 1$. On the other hand, if $Y_{i,j} = 0$, then $-m + 1 \leq t_j - t_i \leq m + 1$, which is never active because m is a large constant. In Section VI, a quantitative comparison of these two methods is presented.

Some additional constraints are required for consistency. An initial and a final POI must be defined, e.g. the entry and exit points. Moreover, all intermediate POIs must have an incoming and an outgoing edge: $p_i = 1 \implies \exists j, h$ s.t. $Y_{j,i} = 1 \wedge Y_{i,h} = 1$, or, equivalently, $p_i = \sum_j Y_{j,i}$, $p_i = \sum_h Y_{i,h}$.

Therefore, the above discussed constraints, except for the probabilistic ones, are linear inequalities and would yield an integer linear programming. However, the presence of probabilistic constraints transforms the problem into a Chance-Constrained Stochastic Programming (CCSP) [22]. These constraints are written as $\mathbb{P}\left(\cdot\right) \geq 1 - \alpha$, where the argument is a linear inequality. They can be expressed as the product of a matrix with a subset of the decision variables, with $1 - \alpha$ the criticality level, inherited from the user profile.

Summarising, the problem can been formulated as a CCSP with a linear objective function, a set of linear inequalities and additional stochastic constraints. Let us uniform the notation in order to have a standard mathematical format for the complete optimisation problem. Let $\boldsymbol{x}$ be the vector of the integer decision variables ($\boldsymbol{p}$, $\boldsymbol{Y}$ and $\boldsymbol{t}$) and let $\boldsymbol{A}$ be the matrix of all the non-stochastic inequalities, then we can write $\boldsymbol{Ax} \leq \boldsymbol{b}$, where $\boldsymbol{b}$ is the vector of right-hand sides. The optimisation problem in canonical form is:

$$\max_{\boldsymbol{x} \in \mathbb{N}^n} \boldsymbol{c}^T \boldsymbol{x} \quad s.t.\ \boldsymbol{Ax} \leq \boldsymbol{b}, \quad \mathbb{P}\left(\tilde{\boldsymbol{D}}\boldsymbol{x} \leq \boldsymbol{v}\right) \geq \boldsymbol{1} - \boldsymbol{\alpha}. \qquad (1)$$

where $\boldsymbol{x} \in \mathbb{N}^n$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{k \times n}$, $\tilde{\boldsymbol{D}} \in \mathbb{R}_+^{N \times n}$ is the matrix that collects the probabilistic properties of nodes and edges, $\boldsymbol{v} \in \mathbb{R}_+^N$, $\boldsymbol{b} \in \mathbb{R}^k$ and $\boldsymbol{\alpha} \in (0,1)^N$. Finally, $\mathbb{P}\left(\tilde{\boldsymbol{D}}\boldsymbol{x} \leq \boldsymbol{v}\right) \geq \boldsymbol{1} - \boldsymbol{\alpha}$ is the term that formalises the non-critical constraints, that might be violated with some probability. As a remark, n is the number of optimisation variables, k the number of deterministic constraints, N the number of probabilistic constraints.

IV. CCSP Solutions of the Problem

The CCSP optimisation problem (1) is an effective modelling tool for optimisation problems under uncertainties [23]. In general, the presence of probabilistic constraints, impairing the use of off-the-shelf numeric solvers, renders CCSP instances challenging to solve [24], [25]. Therefore, we are required to recast the CCSP problem to a different form, to take advantage of the efficient solvers available. Depending on the mathematical properties of the random matrix $\tilde{\boldsymbol{D}}$, different reformulations can be applied. If $\tilde{\boldsymbol{D}}$ has a Gaussian distribution, the problem can be modelled as a Second Order Cone Programming (SOCP). Otherwise, in general, an approximated model of the problem (within an interval of confidence) is generated with a Sample Average Approximation (SAA) approach [22]. Given the heterogeneous nature of the different probabilistic constraints, a solution of the inequality $\tilde{\boldsymbol{D}}\boldsymbol{x} \leq \boldsymbol{v}$ in terms of joint probabilities [25] is not suitable for our application. Therefore, we handle each of the stochastic constraints separately, i.e. $\mathbb{P}\left({\boldsymbol{d}^{(i)}}^T\boldsymbol{x} \leq v^{(i)}\right) \geq 1 - \alpha^{(i)}$, for $i = 1, 2, \ldots, N$, where $\boldsymbol{d}^{(i)}$ is the i^{th} row of $\tilde{\boldsymbol{D}}$ and $\alpha^{(i)}$ is the criticality level associated with constraint i.

A. First Solution: Second Order Cone Programming

When the random vector $\boldsymbol{d}^{(i)}$ is normally distributed, $\boldsymbol{d}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the CCSP (1) can be reduced to a deterministic Second Order Cone Programming with integer decision variables. The continuous relaxation of an integer SOCP model is convex, therefore this kind of problems can be efficiently handled by various solvers (e.g. CPLEX, Gurobi, etc.). Let ${\boldsymbol{d}^{(i)}}^T\boldsymbol{x} \leq v^{(i)}$, if $\boldsymbol{d}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then ${\boldsymbol{d}^{(i)}}^T\boldsymbol{x} - v^{(i)}$ is distributed as $\mathcal{N}(\boldsymbol{\mu}^T\boldsymbol{x} - v^{(i)}, \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x})$, thus

$$\mathbb{P}\left({\boldsymbol{d}^{(i)}}^T\boldsymbol{x} \leq v^{(i)}\right) = \Phi\left(\frac{v^{(i)} - \boldsymbol{\mu}^T\boldsymbol{x}}{\sqrt{\boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x}}}\right), \tag{2}$$

where Φ is the standard Gaussian cumulative distribution function (see the discussion in [26]). We require (2) greater than $1 - \alpha^{(i)}$, yielding:

$$v^{(i)} - \boldsymbol{\mu}^T\boldsymbol{x} \geq \Phi^{-1}\left(1 - \alpha^{(i)}\right) \left|\left|\sqrt{\boldsymbol{\Sigma}}\boldsymbol{x}\right|\right|_2, \tag{3}$$

which represents a quadratic cone constraint. We have thus restated problem (1), for $i = 1, 2, \ldots, N$, as

$$\begin{aligned} \max_{\boldsymbol{x} \in \mathbb{N}^n} \boldsymbol{c}^T\boldsymbol{x} \quad & s.t. \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & v^{(i)} - \boldsymbol{\mu}^T\boldsymbol{x} \geq \Phi^{-1}\left(1 - \alpha^{(i)}\right) \left|\left|\sqrt{\boldsymbol{\Sigma}}\boldsymbol{x}\right|\right|_2. \end{aligned} \tag{4}$$

B. Second Solution: Sample Average Approximation

As mentioned above, the random matrix $\tilde{\boldsymbol{D}}$ may have an unknown or uncommon distribution. In this case, an approximated model for problem (1) can be obtained with a Sample Average Approximation (SAA), see [25]. First, we reformulate (1) in terms of probability of violation of the constraint:

$$\begin{aligned} \max_{\boldsymbol{x} \in \mathbb{N}^n} \boldsymbol{c}^T\boldsymbol{x} \quad s.t. \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & \mathbb{P}\left(\tilde{\boldsymbol{D}}\boldsymbol{x} > \boldsymbol{v}\right) \leq \alpha. \end{aligned} \tag{5}$$

As previously discussed, we should consider the stochastic constraint as a set of individual inequalities, e.g. $\mathbb{P}\left({\boldsymbol{d}^{(i)}}^T\boldsymbol{x} > v^{(i)}\right) \leq \alpha^{(i)}$, for $i = 1, \ldots, N$. Let $\{\hat{\boldsymbol{d}}^{(i,j)}\}_{j=1}^M$ be M independent identically distributed samples of the random vector $\boldsymbol{d}^{(i)}$, and $\hat{p}_M^{(i)}(\boldsymbol{x})$ be the proportion of times that $(\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} > v^{(i)}$, in other words:

$$\hat{p}_M^{(i)}(\boldsymbol{x}) = \frac{1}{M}\sum_{j=1}^{M} \mathbb{1}_{(0,\infty)}((\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} - v^{(i)}), \tag{6}$$

where $\mathbb{1}_{(0,\infty)} : \mathbb{R} \to \{0, 1\}$ is the indicator function, valued 1 if the argument is positive and zero otherwise. The optimisation problem with the M samples $\hat{\boldsymbol{d}}^{(i,j)}$ is

$$\begin{aligned} \max_{\boldsymbol{x} \in \mathbb{N}^n} \boldsymbol{c}^T\boldsymbol{x} \quad s.t. \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & \hat{p}_M^{(i)}(\boldsymbol{x}) \leq \gamma^{(i)}, \end{aligned} \tag{7}$$

for $i = 1, 2, \ldots, N$ and $\gamma^{(i)} \in [0, 1]$ the criticality level of the SAA problem. Inequality $\hat{p}_M^{(i)} \leq \gamma^{(i)}$ is expressed by means of new Boolean slack variables $\boldsymbol{z} \in \{0, 1\}^{N \times M}$, hence the problem (1) is approximated with the SAA formulation, as the following Integer Linear Program:

$$\begin{aligned} \max_{\boldsymbol{x} \in \mathbb{N}^n} \boldsymbol{c}^T\boldsymbol{x} \quad s.t. \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & (\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} - V^{(i,j)}\boldsymbol{z}^{(i,j)} \leq v^{(i)}, \\ & \frac{1}{M}\mathbb{1}^T\boldsymbol{z}^{(i)} \leq \gamma^{(i)}, \end{aligned} \tag{8}$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, M$, where $V^{(i,j)} = \mathbb{1}^T\hat{\boldsymbol{d}}^{(i,j)}$ is the sum of the weights of all the variables, i.e. an upper bound on the weight of the solution. Whenever the constraint is violated, i.e. $(\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} > v^{(i)}$, only $\boldsymbol{z}^{(i,j)} = 1$ makes the inequality $(\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} - V^{(i,j)}\boldsymbol{z}^{(i,j)} \leq v^{(i)}$ hold true. On the other hand, if the constraint is satisfied, then $\boldsymbol{z}^{(i,j)}$ can be set to 0. Finally, the last inequality of (8) enforces the frequency of violations below the criticality level $\gamma^{(i)}$.

Remark 1: As stated in [25], the levels of criticality $\boldsymbol{\gamma} = [\gamma^{(1)}, \ldots, \gamma^{(N)}]^T$ may differ from $\boldsymbol{\alpha} = [\alpha^{(1)}, \ldots, \alpha^{(N)}]^T$. However, when $\boldsymbol{\gamma} = \boldsymbol{\alpha}$, the solution of the SAA approximation (8) converges to the solution of the original CCSP model (1) with probability one as the number of samples M increases. In the literature, to the best of the Authors' knowledge, there is not a rule to choose a proper value for M. In [25], the Authors suggested to choose $\boldsymbol{\gamma} = \boldsymbol{\alpha}/2$ and to solve many instances of problem (8) with different (small) values of M, then keeping the best solution. This approach, indeed, is more efficient than the solution of a single instance of (8) characterised by a large number of samples M. In addition, the Authors suggested for M a value in the range $50 - 120$.

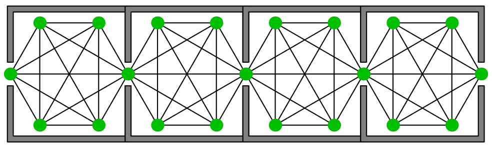


Fig. 3. Portion of a museum, showing the connections between pairs of POIs. Subgraphs within each room are complete.

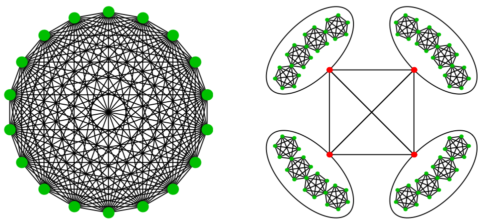


Fig. 4. Left: complete graph (all pairs of POIs are connected). Computationally prohibitive for larger graphs. Right: graph clustering; each block has the same structure of Figure 3.

C. Hierarchical Decomposition and Optimisation

Both the SAA and the SOCP formulations presented above can be solved with good performance by existing solvers for instances of limited size. However, when the number of nodes or edges of the graph becomes too large, the computational times become prohibitive. Therefore, to maintain the computational times acceptable for the considered application, i.e. few tens of seconds at most, larger problems are addressed by decomposing the graph into a number of clusters of nodes, and by employing a hierarchical solution. This clustering can be obtained by grouping POIs belonging to nearby rooms (see Figures 3), or to the same floor, or, more generally, by partitioning and grouping the various nodes based on their relative distances. For each cluster, it is then possible to solve many small SOCP (or SAA) instances, that are thereafter used as building blocks of the optimisation problem to generate the global solution. Once the results for the different clusters are available, their optimal combination representing the global plan can be computed very quickly (tens of milliseconds), as illustrated by numerical experiments in Section VI. Generally, a cluster corresponds to a portion of the environment that the user should visit as a single block. For example, if the planner is applied to generate an activity within a museum, the clustering could partition the POIs according to the floors and to the different topic areas (e.g. plants, mammals, natural history). An example of this scenario is shown in Figure 4, illustrating the complete graph and a possible clustering of its POIs. With this clustering, the full graph is shrunk to a much smaller one, having 4 nodes and 6 links. The objective of the high level problem is to maximise the overall reward gained by visiting some of the POIs, while respecting the user constraints, such as the total available time or the maximum length of the solution. To produce a global plan satisfying all these constraints, a number of different SOCP (or SAA) problems related to each cluster are generated, each with a different allocation of the available resources (e.g. length, time). Therefore, the global plan is produced by finding the best allocation of resources to assign to each cluster, in order to maximise the overall reward. The quality of the plan generated with this approach depends on the number of discretisations of the available resources $N_C \in \mathbb{N}$. A finer discretisation yields better solutions, at the price of increased computational times.

As an example, if the user specifies a maximum duration of 3 hours for the whole activity, we could assign to each cluster a maximum time constraint of 30, 60 and 90 minutes, and determine the optimal reward for each of these scenarios. Since all these optimisation problems are independent from each other, they can be solved in parallel. Moreover, it is also possible to pre-compute the solutions of the different subproblems, and combine the results when necessary to produce user-tailored plans.

The high level hierarchical problem can be formulated as follows. Assuming a partition of the graph into n_χ clusters, we can define the matrices $\boldsymbol{T}, \boldsymbol{D} \in \mathbb{R}^{n_\chi \times N_C}$ of local constraints. Specifically, matrix entries T_{ij} (respectively D_{ij}) represent the j^{th} maximum allowed time (respectively length) assigned to cluster i. Analogously, matrix $\boldsymbol{S} \in \mathbb{R}^{n_\chi \times N_C}$ stores the optimal scores associated with each cluster and each discretisation. Finally, $\boldsymbol{\chi} \in \{0,1\}^{n_\chi \times N_C}$ is the Boolean matrix of unknowns whose entries χ_{ij} represent the visit of cluster i with constraints j. Therefore, the high level hierarchical problem can be expressed as the (deterministic) Integer Linear Program:

$$\begin{aligned} &\max_{\boldsymbol{\chi}} \sum_{i=1}^{n_\chi} \sum_{j=1}^{N_C} S_{ij}\chi_{ij} && s.t. \\ &\sum_{i=1}^{n_\chi} \sum_{j=1}^{N_C} T_{ij}\chi_{ij} \le T_{\max}, && \sum_{i=1}^{n_\chi} \sum_{j=1}^{N_C} D_{ij}\chi_{ij} \le D_{\max}, \\ &\sum_{j=1}^{N_C} \chi_{ij} \le 1 && i = 1, \dots, n_\chi. \end{aligned} \tag{9}$$

The last n_χ constraints are introduced to prevent clusters to be selected more than once. It can be noticed that, in practice, the high-level optimisation problem that we are required to solve with the hierarchical approach has a very small size, and can therefore be solved in a few milliseconds, as discussed in Section VI. A potential issue comes from the existence of a feasible solution for the partitioned problem in case such a solution exists for the whole SAA (the converse is obviously true). In general, the partitioning process could destroy feasible solutions (so much so that the solution eventually found is suboptimal). However, in our setting, the partitioning is dictated by the topology of the problem (e.g, a partition could be a room). Therefore, most of the “interesting” feasible solutions survive the partitioning process and, in our experiment, the suboptimal ends up being quite close to the optimal solution.

V. Motion Planning

The previous sections of the paper focused on the formalisation and solution of the “high-level” Activity Planning problem. Once the sequence of POIs to be visited has been determined, the abstract plan must be refined into an “executable” plan by the Motion Planner, that will be discussed in this section. This requires an algorithm for path planning, but our approach does not need a specific method. We propose

a suitable technique that is effective for our application, but there can be others, as illustrated in the introduction.
Both the length and the user comfort have to be considered for the design of a path by the Motion Planner. While the length is solely a geometric property, the comfort index can be specified via various properties and depends both on the dynamical model of the vehicle and on the shape of the path. These two aspects can be considered together, in a "global" approach. The optimisation problem J, for positive weights w_1, w_2, w_3, is defined as:

$$\min J = w_1 \int_{\mathcal{P}} \mathrm{d}s + w_2 \int_{\mathcal{P}} \mathrm{d}t + w_3 \int_{\mathcal{P}} c(s)\,\mathrm{d}s, \qquad (10)$$

where $\mathcal{P}$ is the path, the first integral minimises the length, the second the total time and the third the comfort function $c(s)$. If v is the speed profile along the path $\mathcal{P}$, integrals with respect to time $\mathrm{d}t$ and space $\mathrm{d}s$ are related with the formula $\mathrm{d}s = \mathrm{v}\,\mathrm{d}t$, thus it is possible to reformulate the previous target functional completely in space or time.
The complexity of the problem can be reduced addressing two subproblems separately: firstly the geometric part, accounting for the shape, the length and the comfort (expressed in terms of minimum jerk, see [20], [27], [28]); secondly the kinematic part, dealing with the vehicle dynamics, and yielding a speed profile v to be tracked along the path. A similar decomposition is discussed in [29], [30], [31]. Following the work of [20], [27] we model the human/walker trajectories as clothoid curves. Thus, the objective function to be minimised depends both on the length of the path and on the total jerk. For an efficient manipulation of clothoid curves we rely on [32], [33]. Within our framework, the adopted planning algorithm is a variation of the traditional RRT* [34], called Informed-RRT* (I-RRT*) [35]. When the algorithm starts, and there is no available solution, it works like RRT*: it samples new points inside the free configuration space and connects them to the closest nodes within the search tree, if the connecting subpath is collision free. After a new node is inserted into the tree, a "rewiring" operation is performed: if passing through the new node reduces the total cost to reach some of its neighbours, their costs and edges are updated accordingly. The main difference between I-RRT* and RRT* is perceivable only after a feasible solution has been found. Indeed, when a solution is available, contrarily to RRT*, I-RRT* restricts the sampling region to the set of points having a heuristically estimated cost lower than the current known optimum.
The modelling of the environment variability is accomplished using statistic information on the areas crossed by the path. The result of the technique proposed here (i.e., a sequence of motion primitives connecting different points) can be used as an input for a dynamic motion planning strategy that adapts the plan to the contingent condition in the field. For instance, the technique proposed in [36] can be used to adapt the motion plan to the presence of humans in the environment. Once, the plan is adapted, it can be implemented through a real–time control strategy. For instance, the authority sharing techniques solutions presented in [37], [38], and extensively validated on elder people in [39], can be a very good fit.

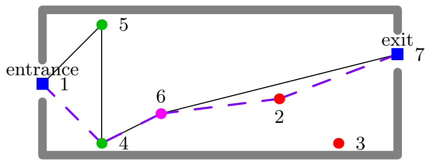


Fig. 5. Toy map of the museum with an entrance and exit point and other POIs that are marked in different colours to represent different categories of interest for the user. The solid line represents the optimal visit sequence for the preference score $\boldsymbol{c} = [0, 2, 2, 10, 10, 5, 0]^T$, the dashed line is the result if the user does not express a preference, i.e. for scores $\boldsymbol{c} = [0, 1, 1, 1, 1, 1, 0]^T$.

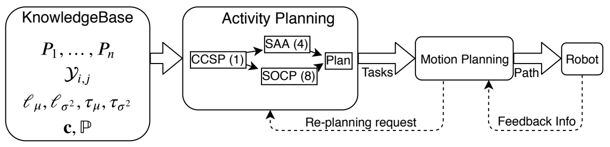


Fig. 6. Block diagram showing the flow among the high-level Activity Planning, the low-level system (Motion Planner, robot, sensing system) and the Knowledge base.

VI. Experiments and Comparisons

This section presents an experimental validation of the proposed Activity Planner. We discuss the application of our methods to synthesis an optimal plan to visit a museum. The validation is done in three steps, firstly, a toy example is discussed, to show how to construct the problem. Then a realistic case, inspired by the museum of natural sciences of our city, is proposed. Finally, an extensive battery of examples is studied: we construct 900 scenarios, divided into small, medium and large sized test cases. Each of them is solved with the eight techniques proposed: SOCP and SAA, solved on the full graph and with the hierarchical strategy, with/out the lazy constraints.

A. A Toy Example

We begin by showing a toy example inspired to a museum visit. The main objective is to produce an activity plan considering the distribution of the distances among 5 POIs and the related travelling times. Each POI is part of some categories, and each category has a rating ranging from 0 to 10, where 10 represents the highest preference.

The visit time for each POI, i.e., the time spent by the user in the desired POI, is chosen constant and equal to 5. The global plan is constrained to have an overall time duration and a total distance travelled lower than $T = 37$ and $L = 17$, respectively (the measurement unit is not really relevant for the abstract formulation of this example). We assume to know the map of the museum, the location of the 5 POIs ($P_2, \ldots, P_6$), and of the entrance and exit points (P_1 and P_7) (see Figure 5 for the topology and Figure 6 for a schematic view of the variables and the flow of the system). Therefore, the total number of points is $n = 7$.

Objective function. The POIs are tagged with a category, which is represented in Figure 5 with a different colour. Each category is associated with a score and the POIs' scores are collected in the vector $\boldsymbol{c}$. The objective is to maximise the total score of the visited POIs, defined as $\boldsymbol{c}^T\boldsymbol{p}$, where

TABLE I
DISTANCE MATRICES ℓ_μ AND ℓ_{σ^2}.

	ℓ_μ							ℓ_{σ^2}						
i \ j	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1	-	4.9	6.2	1.7	1.7	2.5	7.3	-	8.6	13.9	1.1	1.1	2.3	19
2	4.9	-	1.6	3.8	3.9	2.5	2.6	8.6	-	0.9	5.2	5.5	2.2	2.3
3	6.2	1.6	-	4.9	5.4	3.7	2.2	13.9	0.9	-	8.5	10.7	4.9	1.7
4	1.7	3.8	4.9	-	2.4	1.4	6.4	1.1	5.2	8.5	-	2.1	0.7	14
5	1.7	3.9	5.4	2.4	-	2.2	6.1	1.1	5.5	10.7	2.1	-	1.7	13
6	2.5	2.5	3.7	1.4	2.2	-	5.0	2.3	2.2	4.9	0.7	1.7	-	9.1
7	7.3	2.6	2.2	6.4	6.1	5.0	-	19.4	2.3	1.7	14.6	13.5	9.1	-

TABLE II
TIME MATRICES τ_μ AND τ_{σ^2}.

	τ_μ							τ_{σ^2}						
i \ j	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1	-	6.3	8.0	2.2	2.2	3.2	9.4	-	14.2	23.0	1.8	1.8	3.8	32
2	6.3	-	2.1	4.9	5.0	3.2	3.3	14.2	-	1.6	8.6	9.1	3.7	3.9
3	8.0	2.1	-	6.3	7.0	4.8	2.8	23.0	1.6	-	14.1	17.7	8.2	2.9
4	2.2	4.9	6.3	-	3.1	1.8	8.2	1.8	8.6	14.1	-	3.5	1.1	24
5	2.2	5.0	7.0	3.1	-	2.8	7.9	1.8	9.1	17.7	3.5	-	2.9	22
6	3.2	3.2	4.8	1.8	2.8	-	6.5	3.8	3.7	8.2	1.1	2.9	-	15
7	9.4	3.3	2.8	8.2	7.9	6.5	-	32.0	3.9	2.9	24.1	22.3	15	-

$\boldsymbol{p} = [p_1, \dots, p_7]^T$. The scores chosen by an hypothetical user, according to her/his preferences, are $\boldsymbol{c} = [0, 2, 2, 10, 10, 5, 0]^T$. The entrance and exit points (p_1 and p_7) have score and visit time equal to zero because they are not POIs to be visited but still must be part of the plan. This means that the most rewarding POIs are P_4 and P_5 (with a score of 10), that correspond to the integer variables p_4, p_5.

If the user has no particular preference, which can be modelled with equal scores (set to 1 or to any other positive constant, e.g. $\boldsymbol{c} = [0, 1, 1, 1, 1, 1, 0]^T$), then the plan is solely determined by enforcing the constraints. As a consequence, the number of visited POIs will be maximised on the basis of the time required for each visit and of the distance travelled.

Consistency constraints. A first equation encodes the constraint that one of the edges has to emanate from the the start node (the entrance): $\sum_{j=2}^{7} Y_{1,j} = 1$. This is an obvious translation of the constraint, assuming the definition of the Boolean variables $Y_{i,j}$ given in Section III. Likewise, the fact that only one edge reaches the exit node is translated as $\sum_{j=1}^{6} Y_{j,7} = 1$. Notice that a general equality constraint $f(\boldsymbol{x}) = 0$, such as the ones introduced above, can be transformed into the two inequalities $f(\boldsymbol{x}) \leq 0$ and $f(\boldsymbol{x}) \geq 0$, which are consistent with the general form presented in (1). Each visited node should have an incoming edge and an outgoing edge (except the entrance node P_1 and the exit node P_7). This fact produces a set of constraints of the form

$$\left\{ \sum_j Y_{j,i} = p_i \right\}_{i=2}^{7} \cup \left\{ \sum_j Y_{i,j} = p_i \right\}_{i=1}^{6}.$$

To avoid sub-tours, two possibilities exist: with lazy constraints or by keeping track of the node visiting order (as discussed in Section III). In this example, we adopt the second option. Hence, we need the help of another set of integer variables, $\boldsymbol{t}$, of the same dimension of $\boldsymbol{p}$, expressing the visit order. We fix the initial node $t_1 = 1$, then we set for each active edge (i, j) the constraint $t_i < t_j$:

$$\left\{ -m(1-Y_{i,j})+1 \leq t_j - t_i \leq m(1-Y_{i,j})+1 \text{ s.t. } i \neq j \right\}_{i,j=1}^{7}$$

with $m = n + 1$ in this example (in general, it should be greater than the maximum path size n). All these constraints are required to obtain a consistent solution.

User's constraints. The user, for this example, requires a path shorter than a maximum distance L within a probability of failure of $\alpha_l = 0.05$. This gives in Equation (4) a value of $\Phi^{-1}(1 - \alpha_l) \approx 1.64$. For a generic path between POIs P_i and P_j, we denote with $\ell_\mu(i, j)$ and $\ell_{\sigma^2}(i, j)$ the mean and the variance of the corresponding distance, respectively, reported in Table I. So, the mean and the variance of the planned distance will be $\mu_l = \sum_{i \neq j=1}^{7} Y_{i,j} \ell_\mu(i, j)$ and $\sigma_l^2 = \sum_{i \neq j=1}^{7} Y_{i,j} \ell_{\sigma^2}(i, j)$. The constraint thus becomes $L - \mu_l \geq \Phi^{-1}(1 - \alpha_l)\sigma_l$. The constraint on the maximum duration is similar. If T is the maximum allowed time, $\alpha_\tau = 0.05$ is the probability of not satisfying the constraint (implies $\Phi^{-1}(1 - \alpha_\tau) \approx 1.64$), and the means and variances are given in Table II, we obtain $T - \mu_\tau - v \geq \Phi^{-1}(1 - \alpha_\tau)\sigma_\tau$, where $v = \sum_{i=2}^{6} p_i v_t(i)$ is the overall visit time and $v_t = [0, 5, 5, 5, 5, 5, 0]^T$ is the vector of POIs visit times. The vector $\boldsymbol{x}$ of the unknowns of problem (4) is the concatenation of the binary variables $\boldsymbol{p}$ and $\boldsymbol{Y}$ and the integer variables $\boldsymbol{t}$. To standardise the problem, the inequalities are recast into matrix form so that the final formulation becomes (4).

Results. The optimal computed visit sequence for $\boldsymbol{c} = [0, 2, 2, 10, 10, 5, 0]^T$ is P_1, P_5, P_4, P_6, P_7 (see the solid line path in Figure 5), with an overall score of 25, a mean length of $\mu_l = 10.5$ and standard deviation $\sigma_l = 3.6$, while the mean time is $\mu_\tau + v = 28.6$ with $\sigma_\tau = 4.6$. With probability of $1 - \alpha_l = 0.95$, the overall length of the path is less than 16.5 and with probability $1 - \alpha_\tau = 0.95$ the overall time is less than 36.2. It can be noticed that the solution satisfies all the constraints. The POIs P_4 and P_5 have a higher score than the others and are both preferred. On the other hand, even if POI P_2 is on the way, it is not part of the plan because the visit time required would cause a constraint violation. In the case the user does not specify a particular preference for the POIs, i.e. $\boldsymbol{c} = [0, 1, 1, 1, 1, 1, 0]^T$, the result of the optimisation is P_1, P_4, P_6, P_2, P_7 (see the dashed path in Figure 5). With probability 0.95, the length of the path is less than 12.2 and the total visit time less than 30.7. The difference between the two solutions depends on the scores of the POIs: in the first case P_5 is very rewarding with respect to P_2, while in the second all the scores are equal, therefore every sequence of five POIs satisfying all the constraints is an optimal solution, and the solver can return any of them. The toy example is solved in a few milliseconds using Julia and the Gurobi optimiser.

B. Realistic Scenario

A realistic scenario for the Activity Planner is now considered. We adapted the map of an existing museum (i.e.,

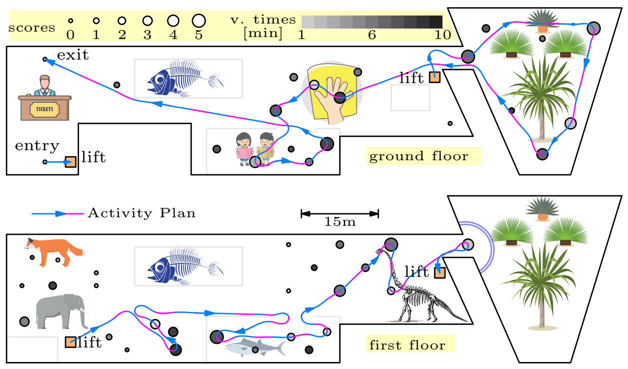


Fig. 7. Map of the ground and first floor of Muse. The blue/magenta solid line is the synthesis of the Activity Planner, the direction is specified by the arrows. The orange squares are the lifts that connect the floors. The thematic areas are categorised with a representative logo, for instance, plants in the greenhouse, a dinosaur for the fossils and so on. The POIs are shown as circles, with the radii proportional to the scores, from 0 (no interest) to 5 (high interest); the filling colour represents the visiting time, from a pale colour (less time) to dark colour (long visit), see also the corresponding colormap.

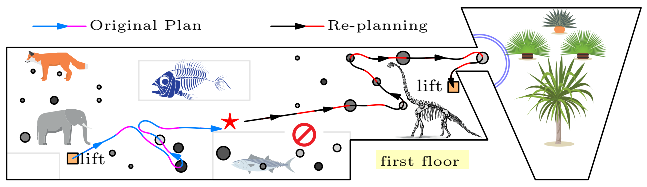


Fig. 8. Deviation from the original plan (Figure 7) due to the unforeseen closure of the fish area.

the MUSE in the city of Trento, Italy), that is composed of different thematic areas (i.e. different POIs tags). The museum develops over two floors that are connected with lifts, see Figure 7 for the layout, with the thematic zones and the relative POIs highlighted. The user has given scores to each POI, with importance proportional to the depicted radius. The estimated visit time is represented by the filling colour of the circle, as detailed in the legend and in the caption of Figure 7. The synthesis of a plan is reported, which was obtained accounting for a maximum distance constraint of 1000 meters, together with a time limit of two hours, both to be respected within 95% probability. The activity begins at the museum's entry point at the ground floor, followed by taking the lift (orange square) to reach the upper floor. The first thematic area visited is the mammals, but more time is spent for fishes and dinosaurs. Before returning by lift to ground floor, there is a visit to the panoramic terrace over the conservatory. Then, the plan proceeds visiting some of the POIs within the tropical greenhouse. Notice that some POIs are skipped even if they are close to the path, due to their high visit time and low scoring. The last part of the plan brings the user to the sensorial zone and to the area dedicated to kids, before ending the visit. We discuss for this example only the SOCP solution, for the sake of brevity. A full comparison of all the eight proposed solution strategies on a large number of scenarios is presented in the next section. The computational time for this example scenario is about 22 seconds, with convergence to the optimal solution. The travelled distance is 482 meters within 95% probability, while the duration is few seconds below the limit of two hours within 95%, thus the solution is dominated by the time constraint.

To illustrate a realistic example of a possible re-planning request, e.g. to overcome an unforeseen situation as discussed at the end of Section II, suppose now that the designed visit to the fish area cannot be carried out due to a non-programmed maintenance intervention (see red star in Figure 8). Since the original solution has become unfeasible, a new instance of the Activity Planner is run, in order to get a new plan (shown with the dashed black and red line in Figure 8). The new plan includes an additional POI in the fossils area on the first floor, while the rest of the plan stays unaltered. The re-planned solution is computed in about 5 seconds.

C. Experimental Validation

To validate the proposed activity planner, we extend the results presented in the previous examples, by conducting 900 experiments under different conditions, based on three different graph layouts, classified as small, medium and large sized. In order to validate the hierarchical technique described in Section IV, each layout is composed of different blocks, representing specific physical areas (floors and sections of the museum), each including a collection of rooms filled with POIs. An example of the considered graph layout pattern is shown in Figures 3 and 4. The subgraph within each room is complete, meaning that the POIs of a room are all connected, while pairs of rooms are linked through entry and exit points. For each execution, all the parameters associated with each POI and action, the bounds on the maximum duration of the visit and maximum travelled distance, and all the data required by the Activity Planner are randomly generated. We call the ensemble graph, probabilistic distributions of visit times, distances, travel times, etc. a *scenario*. To validate the proposed approach, each scenario is solved with all the eight techniques presented (see Table III), namely: the SOCP and SAA problems, solved completely (full problem) and with the hierarchical decomposition (identified with "_h"). The numerical tests are conducted with and without the lazy constraints (the latter identified with "_ns", i.e. without sub-tour elimination constraints), as described in Section III. For each of the three museum sizes (small, medium and large), 300 scenarios are considered, each comprising a different graph and different parameters. For the small museums we have 4 clusters, 4 rooms per cluster and 4 POIs per room; for the medium sized we have 5 clusters, 5 rooms per cluster and 5 POIs per room; finally, the large sized museums have 5 clusters, 8 rooms per cluster and 10 POIs per room. Hence, the total number of POIs is respectively 64, 125 and 400. Actually, for our purposes, typical real settings have dimensions ranging from small to medium sized scenarios, up to one hundred POIs. However, we test also larger scenarios to show the scalability and applicability of the proposed approach in general and the effectiveness of the hierarchical decomposition. The parameters for each museum are chosen randomly: the random distribution of scores and visit times is uniform, whereas travel times and distances are distributed normally.

For the hierarchical solution, the number N_C is partitioned as follows: in Table III we have $N_C = 9 = 3 \times 3$, that is 3 values for the time and 3 values for the distance, Table IV shows a finer granularity, $N_C = 30 = 6 \times 5$. As highlighted in Section IV-C, the finer the granularity, the better the solution, but at the cost of an increased computational time; this can be seen comparing Tables III and IV. The optimal solution for each combination of local constraints is obtained from the solution of the CCSP problem (1), that can be solved both via SOCP (4) or via SAA (8). The tuning parameters for the algorithms are four: N_C, $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$ and M. The number of discretisations N_C can be chosen according to the available computational resources. As a rule of thumb, it is possible to start with a coarse splitting and then refine while the elapsed time is below an user defined threshold value. The criticality levels $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ tune the probability of not satisfying a probabilistic constraint, thus, they do not alter the computational times. The criticality level $\boldsymbol{\alpha}$, in general, is chosen on the basis of the user profile; in our experiments the same value is chosen for all the constraints, i.e. $\boldsymbol{\alpha} = \alpha\mathbb{1}$ for $\alpha = 0.05$. Based on Remark 1, we select $\boldsymbol{\gamma} = \boldsymbol{\alpha}/2$. Moreover, for the parameter M, that should be chosen in the range $50-120$, we have determined experimentally that a value of 75 provides a good trade-off between accuracy (w.r.t. SOCP equivalent) and performance. Therefore, for all the examples, we generate a number $M = 75$ of samples (from an estimated Gaussian distribution), each modelling a different circumstance.

The hierarchical method firstly solves the $n_\chi \times N_C$ (i.e. number of clusters times number of different discretisations) SOCP/SAA subproblems, obtaining the optimal scores $\boldsymbol{S}$. The second step is to find the solution to the high level problem (9), thus obtaining the overall (suboptimal) solution. We employed Gurobi through the Julia interface to solve the Integer Program. Being the problem modelled as a standard optimisation, we could use any off-the-shelf tool available, inheriting its properties in terms of completeness, complexity and computational performance. We report the solving times (and standard deviations in brackets) in Tables III and IV (they include both the optimisation time, and the time required to build the model given as input to the solver). A timeout of 30 seconds is set in order to limit the computational time of each hierarchical subproblem, after that, the best (feasible) solution found so far (if any) is returned. The solution of the high-level problem for the hierarchical approach takes only tens of milliseconds, therefore it does not affect the total time, dominated by the solution of the low level subproblems. The solver timeout for the optimisation of the full problems without the hierarchical decomposition, is set to 400 seconds. As shown in Table III, this timeout affects only the solutions without the lazy constraints. Indeed, with the adoption of the lazy sub-tour elimination constraints, the solver is able to converge to the global optimum before the timeout.

D. Discussion of the results

To measure the quality of the computed solutions in the cases where we are not able to find the global optimum for the complete problem, it is possible to compare the found suboptimal solution with the optimal solution of its continuous relaxation. Indeed, the convexity of the relaxed problem, both for the SAA (linear program) and for the SOCP (second-order cone programming) formulation, leads to the global optimum. Since the integral constraints on the decision variables make the problem non-convex, the relaxed solution provides a conservative upper bound on the optimal score, that can be used as the quantitative measure of the quality of the found solution. More in details, the solver produces a sequence of relaxed solutions, yielding a non-increasing sequence of upper bounds on the optimal value. The solver stops either when a time-out is triggered, or if the gap between the upper bound and the current (feasible) solution becomes zero. In our experiments, the optimum is always found before 30 seconds for both the small and medium sized scenarios, as illustrated in Table III, where the scores are normalised to 100% (corresponding to the value of the optimal solution).

Only for the large scenarios the time-out stops the solver for some of the sub-problems of the hierarchical approach, however the gap between the best solution found and the upper bound provided by the relaxation is on average smaller than 15%. The overall difference between the hierarchical approximation and the global optimal solution (which, for large scenarios, is not a practicable way due to excessive computational times) is found to be less than 20% from the upper bound, as can be seen from Table III and Table IV. Therefore, from a practical point of view, the quality of the solution is really satisfactory; moreover, the number of 400 POIs of the large scenarios is way beyond the typical use case of our application. In addition, if the found solution were not satisfactory, it would always be possible to trigger a second optimisation, either by resorting to a multi-start approach (i.e., starting the solver from a different initial guess), by modifying the parameters of the solver (tolerances) or by changing solution strategy.

The analysis of the experiments leads us to the following comments: the SOCP method is in general faster than the SAA method for the hierarchical subproblems; it is possible to obtain the same result of the SOCP with a SAA approximation with 75 samples; even with a small number of discretisations N_C, it is possible to get good suboptimal solutions; with the hierarchical technique it is possible to handle, in a reasonable amount of time, large problems that cannot be tackled directly (see the column relative to large sized scenarios In Table IV).

Comparison with conventional methods To further substantiate the effectiveness of our approach, we report here a comparison with standard techniques. As aforementioned, these methods can only handle one stochastic constraint or multiple deterministic constraints, therefore a common practice is to convert the probabilistic bounds into deterministic ones. This is achieved by replacing each random variable with the equivalent worst case percentile. For instance, if the mean length of an edge is $\mu_l = 10$, the standard deviation $\sigma_l = 2$, the probability of failure $\alpha_l = 0.05$ (as in the carried out examples), the converted deterministic length is $\mu_l + \Phi^{-1}(1-\alpha_l)\sigma_l = 10 + 1.64 \cdot 2 = 13.28$, where Φ is the standard Gaussian cumulative distribution function. A possible drawback of this conversion is that it could end up producing very conservative results, being based on worst case scenarios.

TABLE III
RESULTS FOR THE THREE SIZES (900 SCENARIOS), MEAN TIMES IN SECONDS, STD DEVIATION IN BRACKET, SCORE IN % W.R.T. BEST FOUND.

	small sized		medium sized		large sized	
Problem	time	score	time	score	time	score
saa	6 (1)	100	14 (5)	100	355 (55)	100
saa_h	21 (3)	77	66 (8)	72	320 (40)	80
saa_ns	20 (10)	100	111 (59)	100	426 (82)	98
saa_ns_h	30 (3)	77	108 (14)	72	990 (253)	80
socp	5 (4)	100	21 (11)	100	344 (55)	100
socp_h	10 (1)	77	31 (4)	72	191 (31)	80
socp_ns	19 (14)	100	104 (57)	100	413 (73)	91
socp_ns_h	21 (3)	77	80 (15)	72	652 (200)	80

TABLE IV
RESULTS FOR THE THREE SIZES (900 SCENARIOS) WITH A FINER DISCRETISATION ($N_C = 30$), MEAN TIMES IN SECONDS, STD DEVIATION IN BRACKET, SCORE IN % W.R.T. BEST FOUND.

	small sized		medium sized		large sized	
Problem	time	score	time	score	time	score
saa_h	75 (8)	96	260 (24)	96	1633 (84)	82
saa_ns_h	104 (10)	96	394 (42)	96	2922 (341)	82
socp_h	34 (4)	96	118 (15)	96	1259 (49)	82
socp_ns_h	70 (8)	96	274 (36)	96	2245 (231)	82

The comparison with this deterministic approach is carried out on the toy example of Section VI-A with POIs scores $\boldsymbol{c} = [0, 2, 2, 10, 10, 5, 0]^T$ and keeping the same parameters. The visit sequence obtained with the deterministic approach is P_1, P_4, P_6, P_2, P_7, which avoids the POI P_5 (the most rewarding) due to a constraint violation, while the sequence for our approach is P_1, P_5, P_4, P_6, P_7. The length and time (with probability 0.95) using the deterministic approach are both lower than those obtained with our approach, but the score is lower as well, i.e. 17 w.r.t. 25. This is a further evidence that managing stochastic constraints in place of deterministic ones is beneficial, even for small sized problems. In fact, this gap becomes more evident when the size of the problem and the number of POIs composing the solution increase.

VII. CONCLUSIONS

Planning activities tailored to different users, maximising their satisfaction while respecting all the given physical constraints is a key aspect for a robotic walker used to assist elderly throughout the performance of social activities in public spaces. The solution presented in this paper is based on a strong interaction between the "high level" and the "low level" planning components (i.e. the Activity Planner and the Motion Planner). With the support of the Motion Planner, the physical parameters associated with the different motion actions are determined. Moreover, the Motion Planner refines an activity plan to an actual sequence of paths that the assistive robot can follow. The Activity Planner, on the other hand, selects the optimal sequence of motion actions composing the plan by solving a CCSP problem, that models all the deterministic and probabilistic constraints deriving from the user and the environment. In addition, we propose a hierarchical approach, that can be used to solve instances on larger environments within a few tens of seconds (a reasonable time-frame for this application), producing very good (suboptimal) solutions.

Several possible research directions lie before us. First, we will seek different solution algorithms for the Activity Planner, in order to reduce the execution time and/or improve the quality of the suboptimal solution. We will also investigate different models in which the user requirements could be associated with a level of importance, which could be distinct from the level of criticality considered in this paper. For instance, a requirement could be stated in probabilistic terms (e.g., for physical limitations) but it could be more important for the user than other deterministic requirements in a situation in which they cannot be all satisfied. The development of a metaheuristic solution capable of solving the complete Activity Planning problem to find sub-optimal solutions in a reasonable amount of time, and the study of its application in combination with or as an alternative to the Integer Programming solvers, is an interesting research topic and is deferred to future work. Finally, we will explore the application of our framework in other areas of robotics, such as automated delivery in crowded environments (such as historic city centres).

REFERENCES

[1] H. Liu, "Exploring human hand capabilities into embedded multifingered object manipulation," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 389–398, 2011.

[2] L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, S. S. Cash, P. van der Smagt *et al.*, "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm," *Nature*, vol. 485, no. 7398, p. 372, 2012.

[3] L. Lunenburger, G. Colombo, R. Riener, and V. Dietz, "Clinical assessments performed during robotic rehabilitation by the gait training robot lokomat," in *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on.* IEEE, 2005, pp. 345–348.

[4] B. Graf, M. Hans, and R. D. Schraft, "Care-O-bot IIdevelopment of a next generation robotic home assistant," *Autonomous robots*, vol. 16, no. 2, pp. 193–205, 2004.

[5] O. Chuy, Y. Hirata, and K. Kosuge, "Control of walking support system based on variable center of rotation," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2289–2294.

[6] K.-T. Yu, C.-P. Lam, M.-F. Chang, W.-H. Mou, S. H. Tseng, and L. C. Fu, "An interactive robotic walker for assisting elderly mobility in senior care unit," in *2010 IEEE Workshop on Advanced Robotics and its Social Impacts*, Oct 2010, pp. 24–29.

[7] L. Palopoli *et al.*, "Navigation assistance and guidance of older adults across complex public spaces: the DALi approach," *Intelligent Service Robotics*, vol. 8, no. 2, pp. 77–92, Apr 2015.

[8] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Snchez-Ante, "Planning tours of robotic arms among partitioned goals," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 207–223, 2006.

[9] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, June 2010.

[10] M. Van Den Briel, R. Sanchez, M. B. Do, and S. Kambhampati, "Effective approaches for partial satisfaction (over-subscription) planning," in *AAAI*, 2004, pp. 562–569.

[11] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *EU. J. of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016.

[12] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle, "Metaheuristics for the bi-objective orienteering problem," *Swarm Intelligence*, vol. 3, no. 3, pp. 179–201, 2009.

[13] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems," *Journal of Heuristics*, vol. 20, no. 3, pp. 291–328, 2014.

[14] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[15] N. Ganganath *et al.*, "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 601–611, 2015.

[16] A. Colombo, D. Fontanelli, A. Legay, L. Palopoli, and S. Sedwards, "Efficient customisable dynamic motion planning for assistive robots in complex human environments," *Journal of ambient intelligence and smart environments*, vol. 7, no. 5, pp. 617–634, 2015.

[17] J. J. Kim and J. J. Lee, "Trajectory optimization with particle swarm optimization for manipulator motion planning," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 620–631, June 2015.

[18] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Trans. on robotics and automation*, vol. 16, no. 5, pp. 615–620, 2000.

[19] A. Furieri, "SpatiaLite," www.gaia-gis.it/fossil/libspatialite/index, 2015.

[20] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz, "An Optimality Principle Governing Human Walking," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 5–14, Feb 2008.

[21] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press, 2007.

[22] J. Luedtke and S. Ahmed, "A sample approximation approach for optimization with probabilistic constraints," *SIAM Journal on Optimization*, vol. 19, no. 2, pp. 674–699, 2008.

[23] M. Zare, T. Niknam, R. Azizipanah-Abarghooee, and A. Ostadi, "New stochastic bi-objective optimal cost and chance of operation management approach for smart microgrid," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 6, pp. 2031–2040, Dec 2016.

[24] G. C. Calafiore and L. E. Ghaoui, "On distributionally robust chance-constrained linear programs," *Journal of Optimization Theory and Applications*, vol. 130, no. 1, pp. 1–22, Jul 2006.

[25] B. K. Pagnoncelli, S. Ahmed, and A. Shapiro, "Sample average approximation method for chance constrained programming: Theory and applications," *Journal of Optimization Theory and Applications*, vol. 142, no. 2, pp. 399–416, Aug 2009.

[26] E. Camponogara, A. B. de Oliveira, and G. Lima, "Optimization-based dynamic reconfiguration of real-time schedulers with support for stochastic processor consumption," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 594–609, Nov 2010.

[27] P. Bevilacqua, M. Frego, E. Bertolazzi, D. Fontanelli, L. Palopoli, and F. Biral, "Path planning maximising human comfort for assistive robots," in *2016 IEEE Conference on Control Applications (CCA)*. IEEE, 2016, pp. 1421–1427.

[28] E. Bertolazzi and M. Frego, "Interpolating clothoid splines with curvature continuity," *Mathematical Methods in the Applied Sciences*, vol. 41, no. 4, pp. 1723–1737, 2017.

[29] E. Velenis and P. Tsiotras, "Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation," *J. of Opt. Theory and App.*, vol. 138, no. 2, pp. 275–296, 2008.

[30] M. Frego, P. Bevilacqua, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, "Trajectory planning for car-like vehicles: A modular approach," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 203–209.

[31] M. Frego, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, "Semi-analytical minimum time solutions with velocity constraints for trajectory following of vehicles," *Automatica*, vol. 86, pp. 18 – 28, 2017.

[32] E. Bertolazzi and M. Frego, "G1 fitting with clothoids," *Mathematical Methods in the Applied Sciences*, vol. 38, no. 5, pp. 881–897, 2015.

[33] E. Bertolazzi and M. Frego, "On the G^2 Hermite interpolation problem with clothoids," *J. of Computational and Applied Mathematics*, vol. 341, pp. 99 – 116, 2018.

[34] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2011, pp. 1478–1483.

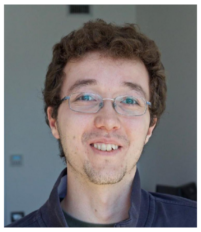
[35] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Int. Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, 2014, pp. 2997–3004.

[36] P. Bevilacqua, M. Frego, D. Fontanelli, and L. Palopoli, "Reactive planning for assistive robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1276–1283, April 2018.

[37] M. Andreetto, S. Divan, F. Ferrari, D. Fontanelli, L. Palopoli, and F. Zenatti, "Simulating Passivity for Robotic Walkers via Authority-Sharing," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1306–1313, April 2018.

[38] M. Andreetto, S. Divan, F. Ferrari, D. Fontanelli, L. Palopoli, and D. Prattichizzo, "Combining Haptic and Bang-Bang Braking Actions for Passive Robotic Walker Path Following," *IEEE Transactions on Haptics*, pp. 1–1, 2019.

[39] F. Ferrari, S. Divan, C. Guerrero, F. Zenatti, R. Guidolin, L. Palopoli, and D. Fontanelli, "Human–Robot Interaction Analysis for a Smart Walker for Elderly: The ACANTO Interactive Guidance System," *International Journal of Social Robotics*, Jun 2019.



Paolo Bevilacqua Paolo Bevilacqua is a Postdoctoral Researcher at the Embedded Electronics and Computing Systems (EECS) Group of the Department of Information Engineering and Computer Science (DISI), University of Trento, Italy, where he obtained his masters degree in computer science in 2015, and his PhD in 2019. His research interests focus mainly on motion planning for wheeled robots, and on socially compliant autonomous navigation in environments shared with humans.

Marco Frego Marco Frego received the M.S. in Mathematics in 2010 and the PhD in Mechatronics in 2014, both from the University of Trento, Italy. After an experience as an assistant professor from 2014 to 2015 at the Hamburg University of Technology (TUHH), Hamburg, Germany, he moves to the Dep. of Inf. Eng. and Computer Science (DISI), and he teaches numerical analysis at the Dep. of Industrial Engineering (DII) of the University of Trento, Italy. Since 2020 he is with the Faculty of Science and Engineering of the Free University of Bolzano. His research interests include optimisation and optimal control, clothoid curves for planning of autonomous and assistive robots.

Luigi Palopoli Luigi Palopoli is Professor of Robotics and Real–Time Embedded systems at the University of Trento. He received his PhD degree from the Scuola Superiore S. Anna in Pisa in 2002. His past research revolved around real–time control systems and control with computation and communication constraints. Recently, he has shifted his interest toward Robotics and he coordinated two large european initiatives in the area of robots for elderly assistance. He is associate editor of the IEEE Transactions on Automatic control and of the Elsevier Journal of Systems Architecture.

Daniele Fontanelli Daniele Fontanelli (M'10, SM'19) received the M.S. degree in Information Engineering in 2001, and the Ph.D. degree in Robotics in 2006, both from the University of Pisa, Italy. He was a Visiting Scientist with the Vision Lab of the University of California at Los Angeles, US, from 2006 to 2007. After a period with the Interdepartmental Research Center "E. Piaggio", University of Pisa, he joined as an Associate Researcher the University of Trento, Italy, where he is now an Associate Professor. He has authored and co-authored more than 150 scientific papers in peer-reviewed top journals and conference proceedings. He is currently an Associate Editor for the IEEE Transactions on Instrumentation and Measurement and for the IET Science, Measurement & Technology Journal. His research interests include distributed estimation and control, human localization algorithms, real-time estimation and control, wheeled mobile robots' estimation and control, and service robotics.