Trajectory Planning

⇒ The trajectory planning problem is to find control (force) input $u(t)$ yielding a trajectory $q(t)$ that avoids obstacles, takes the system to the desired goal state"

- → Optimize some objective function while doing so.
- → This can be considered a complete "motion-planning" problem, as opposed to a "path-planning" problem.

⇒ There are two comon approaches to trajectory Planning for dynamic system.

1. → **Decoupled approach**
   - → first search path in the configuration space
   - → Then find optimal time scaling for the path subjected to actuator limits.
2. → **Direct approach**
   - → Search ~~for a~~ take place in the system's state space.

⇒ In this chapter we focus on fully actuated system

★ Preliminaries

⇒ A path $q(s)$ is assumed to be a twice-differentiable curve on the configuration space Q, $q: [0,1] \to Q$.

⇒ A time scaling $s(t)$ is a function $s: [0, t_f] \to [0,1]$ assigning an s value to each time $t \in [0, t_f]$.

⇒ Together, a path and a time scaling specify a trajectory $q(s(t))$.

- The time scaling $s(t)$ should be twice differentiable & monotonic.
- The twice-differentiability of $s(t)$ ensures that the acceleration $\ddot{q}(t)$ is well defined and bounded.

★ Decoupled Trajectory Planning

⇒ Given a collision-free path $q(s)$ for a robot system, what is the fastest feasible trajectory that follows this path given the actuator constraints?

⇒ Let robot be subjected to actuator limits

$$u_i^{min}(q, \dot{q}) \leq u_i \leq u_i^{max}(q, \dot{q})$$

⇒ Simplest example of actuator limits are the symmetric state-independent bounds

$$|u_i| \leq u_i^{max}$$

⇒ For a given path $q_r(s)$,

$$\dot{q} = \frac{dq}{ds}\dot{s} \quad \text{—— ①}$$

$$\ddot{q} = \frac{d^2q}{ds^2}\dot{s}^2 + \frac{dq}{ds}\ddot{s} \quad \text{— ②}$$

$$u = M(q)\ddot{q} + \dot{q}^T \Gamma(q)\dot{q} + g(q) \quad \text{— ③}$$

{dynamics of robot in standard form}

⇒ Substituting eqn ③ with eqn ① & ②:

$$\left(M(q_r(s))\frac{dq}{ds}\right)\ddot{s} + \left(M(q_r(s))\frac{d^2q}{ds^2} + \left(\frac{dq}{ds}\right)^T \Gamma(q_r(s))\frac{dq}{ds}\right)\dot{s}^2 + g(q_r(s)) = u$$

$$\boxed{a(s)\ddot{s} + b(s)\dot{s}^2 + c(s) = u} \quad \text{— ④}$$

{defines dynamics constrained to the path $q_r(s)$}

$a(s)$ → Inertia product

$b(s)$ → Velocity product

$c(s)$ → gravitational term

⇒ As the robot travels along the path $q(s)$, its state at any time is identified by $(s, \dot{s})$.

⇒ So at any time the system must satisfy the constraints.

$$u^{min}(s,\dot{s}) \leq a(s)\ddot{s} + b(s)\dot{s}^2 + c(s) \leq u^{max}(s,\dot{s}) \quad \text{—⑤}$$

⇒ Let $L_i(s,\dot{s})$ and $U_i(s,\dot{s})$ be the maximum and minimum accelerations $\ddot{s}$ satisfying the i^{th} component of eqn ⑤.

Let $\alpha_i(s,\dot{s}) = \dfrac{u_i^{max}(s,\dot{s}) - b_i(s)\dot{s}^2 - c_i(s)}{a_i(s)}$

Let $\beta_i(s,\dot{s}) = \dfrac{u_i^{min}(s,\dot{s}) - b_i(s)\dot{s}^2 - c_i(s)}{a_i(s)}$

$U_i(s,\dot{s}) = \alpha_i(s,\dot{s})$ & $L_i(s,\dot{s}) = \beta_i(s,\dot{s})$ if $a_i(s) > 0$

& $U_i(s,\dot{s}) = \beta_i(s,\dot{s})$ & $L_i(s,\dot{s}) = \alpha_i(s,\dot{s})$ if $a_i(s) < 0$

⇒ We define

$$L(s,\dot{s}) = \max_{i \in 1,\dots n_q} L_i(s,\dot{s}) \quad \text{—⑥}$$

$$U(s,\dot{s}) = \min_{i \in 1,\dots n_q} U_i(s,\dot{s}) \quad \text{—⑦}$$

⇒ The actuation limit in eq ⑤ can be stated as:

$$\boxed{L(s,\dot{s}) \leq \ddot{s} \leq U(s,\dot{s})} \quad -⑧$$

⇒ Restatement of the problem:

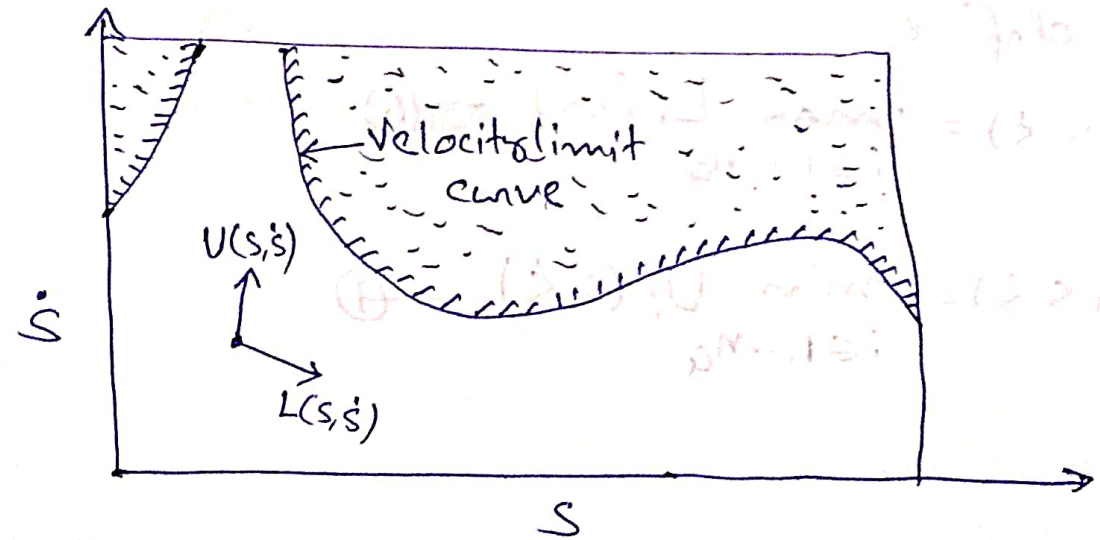
Given a path $q: [0,1] \to Q$, an initial state $(0, \dot{s}_0)$, and a final state $(1, \dot{s}_f)$, $\dot{s}_0, \dot{s}_f \geq 0$, find a monotonically increasing twice-differentiable time scaling $s: [0, t_f] \to [0,1]$ such that

① → Satisfies,

$$s(0) = 0 \qquad s(t_f) = 1$$
$$\dot{s}(0) = \dot{s}_0 \qquad \dot{s}(t_f) = \dot{s}_f$$

② → Minimize the total travel time t_f along the path while respecting the actuation constraints for all time $t \in [0, t_f]$

⇒ We can conveniently visualize this problem in the $(s, \dot{s})$ state space.



⇒ At any state $(s, \dot{s})$, the constraints eq. (8) specify a range of feasible accelerations along the path.

↳ This range can be interpreted as a cone of tangent vectors in the state space.

⇒ The problem is to find a curve from $(0, \dot{s}_0)$ to $(1, \dot{s}_f)$ such that $\dot{s} > 0$ everywhere and the tangent at each state lies inside the cone at that state.

↳ Further, the curve should maximize the speed $\dot{s}$ at each s to minimize the time of motion.

⇒ As consequence of this is that the curve always follows the upper or lower bound of the cone at each state.

↳ This kind of trajectory is called "bang-bang" trajectory.

↳ The heart of the time-scaling problem is to find the switching point between maximum & minimum acceleration.

⇒ At some state $(s, \dot{s})$, the actuation constraints indicate that there is no feasible acceleration that will allow the system to continue to follow the path.

↳ We call these regions inadmissible regions.

⇒ At any inadmissible state, the robot is doomed to leave the path immediately.

⇒ At admissible state, the robot may still be doomed to eventually leave the path.

⇒ We will assume that, for any s, the robot is strong enough to maintain its configuration statically, so all $\dot{s}=0$ states are admissible.

↳ and the path can be executed arbitrarily slow.

⇒ Velocity limit curve $V(s)$ satisfies

$$L(s,\dot{s}) = V(s,\dot{s})$$

⇒ The velocity limit $V(s)$ is obtained by equating $L_i(s,\dot{s}) = V_j(s,\dot{s})$ for all $i,j = 1,\dots n_a$ and solving each equation for $\dot{s}$ (if a solution exists).

↳ For all i,j, keep the minimum value:

$$V(s) = \min_{i,j} \dot{s}_{i,j}(s)^2$$

⇒ The following algorithm uses numerical integration to find the set of switches, expressed as the s value at which the switch occurs

Time-Scaling Algorithm

1. Initialize an empty list of switches $\mathcal{S} = \{\}$ and and a switch counter $i = 0$.
   - Set $(s_i, \dot{s}_i) = (0, \dot{s}_0)$

2. Integrate the equation $\ddot{s} = L(s, \dot{s})$ backward in time $(1, \dot{s}_f)$ until the velocity limit curve is penetrated or $s = 0$ is reached.
   - There is no solution to the problem if $\dot{s} = 0$ is reached.
   - Otherwise call this phase plane curve F and proceed to the next step

3. Integrate the equation $\ddot{s} = U(s, \dot{s})$ forward in time from $(s_i, \dot{s}_i)$.
   - Call this curve A_i.
   - Continue integration until either A_i crosses F or A_i penetrates the velocity limit curve.
   - If A_i crosses F, then increment i, let s_i be the s value at which the crossing occurs:
     - Append s_i to the list of switches $\mathcal{S}$.
     - The problem is solved & $\mathcal{S}$ is the solution.
   - If instead the velocity limit curve is penetrated, let $(s_{lim}, \dot{s}_{lim})$ be the point of penetration and proceed to the next step.

4. Search the velocity limit curve $V(s)$ forward in s from $(s_{lim}, \dot{s}_{lim})$ until finding the first point where the feasible acceleration is tangent to the velocity limit curve.

→ If the velocity limit is $V(s)$, then a point $(s_0, V(s_0))$ satisfies the tangency condition if

$$\frac{dv}{ds}\Big|_{s=s_0} = U(s_0, V(s_0)) / V(s_0)$$

→ Call the first tangent point reached $(s_{tan}, \dot{s}_{tan})$

→ From $(s_{tan}, \dot{s}_{tan})$, integrate the curve $\ddot{s} = L(s, \dot{s})$ backward in time until it intersect A_i.

→ Increment i and call this new curve A_i.

→ Let s_i be the s value of the intersection point.

→ This is a switch point from maximum to minimum acceleration.

→ Append s_i to the list S.

5. Increment i and set $(s_i, \dot{s}_i) = (s_{tan}, \dot{s}_{tan})$.

↳ This is a Switch point from minimum to maximum acceleration.

↳ Append s_i to the list S.

↳ Go to Step 3.

★ Zero Inertia Points

⇒ If $a_i(s) = 0$, the force at the i^{th} actuator is independent of $\ddot{s}$, and therefore the i^{th} actuator defines no acceleration constraints.

↳ Instead, it defines directly a velocity constraint:

$$u_i^{min}(s, \dot{s}) \leq b_i(s)\dot{s}^2 + c_i(s) \leq u_i^{max}(s, \dot{s})$$

⇒ Let $\dot{s}_{zip}^{max}(s)$ be the maximum velocity satisfying all K constraints

↳ Let $\dot{s}^{max}(s) = \min(v(s), \dot{s}_{zip}^{max}(s))$

⇒ $\dot{s}^{max}(s)$ is the true velocity curve, generalizing the curve $v(s)$ by allowing for the possibility of Zero inertial points.

⇒ If a point on the velocity curve $\dot{s}^{max}(s)$ is determined by a zero inertia point velocity constraint, then $L(s,\dot{s}) < U(s,\dot{s})$ at this point.

↳ This point is called a **Critical point**.

⇒ If, in addition, either $U(s,\dot{s})$ integrated forward from this point or $L(s,\dot{s})$ integrated backward from this point, would result in immediate penetration of the velocity limit curve, then the point is called **Singular**.

⇒ At singular point $(s_s, \dot{s}^{max}(s_s))$ let

$$\ddot{s}_{tangent} = \dot{s}^{max}(s_s)\frac{d\dot{s}^{max}}{ds}\Big|_{s=s_s}$$

be the acceleration defined be the tangent to the velocity limit curve.

⇒ If the velocity limit curve is not differentiable at singular point, then define

$$\ddot{s}^{+}_{tangent} = \dot{s}^{max}(s_s)\frac{d\dot{s}^{max}}{ds}\Big|_{s=s_s^{+}}$$

$$\ddot{s}^{-}_{tangent} = \dot{s}^{max}(s_s)\frac{d\dot{s}^{max}}{ds}\Big|_{s=s_s^{-}}$$

to be the right & left limits, respectively.

⇒ Then, to prevent penetration of the velocity limit curve, the maximum feasible acceleration at $(s_s, \dot{s}^{max}(s_s))$ is

$$\ddot{s}^{max} = \min\left(\ddot{s}^{+}_{tangent}, U\right)$$

⇒ Similarly, the minimum feasible acceleration is

$$\ddot{s}^{min} = \max\left(\ddot{s}^{-}_{tangent}, L\right)$$

★ Global Time-Optimal Trajectory Planning

"Find time-optimal trajectory between two states, when we are free to choose any collision-free path"

↳ Conceptually, imagine running the time-scaling algorithm on all possible paths between the start and goal.

↳ Then the fastest of these is the global time-optimal trajectory.

★ Direct Trajectory Planning

⇒ Here, we will study methods of planning the trajectory directly in the state space.

⇒ If we are intrested in finding trajectories that optimize some cost function, such as motion time or expended energy Optimal control theory provides necessary conditions on the trajectories.

- → Unfortunately, these conditions are complex for almost any robot system and cannot be solved analytically.
- → Because of this, we consider two numerical approaches:
  - → nonlinear optimization
  - → grid-based search.

⇒ First we describe how to transform the optimal control problem to a finite-dimensional Parameter optimization problem.

★ Optimal Control

⇒ Given a dynamic system

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u \quad \text{—— (a)}$$

we would like to find a solution $(q(t), u(t))$, $t \in [0, t_f]$ to eq (a) that avoids obstacles & joint limits, respects actuator limits, and takes the system from the initial state $(q_{start}, \dot{q}_{start})$ at time $t=0$ to the final state $(q_{goal}, \dot{q}_{goal})$ at time $t = t_f$.

↳ Of all the trajectories that accomplish this, we might want to find one that minimizes some objective function J.

$$\{ J = J(u(t), q(t), t_f) \}$$

⇒ Let's express the state of the system as

$$x = [q^T, \dot{q}^T]^T$$

⇒ Lets rewrite the equations of motion in the general form:

$$\dot{x}(t) = f(x(t), u(t), t) = f(x, u) \quad \text{—— (b)}$$

{ ∀ state eq that do not change with time }

{ Where f is the vector differential equation describing the kinematic & dynamics of the system }

⇒ Let the objective function J be written

$$J = \int_0^{t_f} \mathcal{L}(x(t), u(t), t)\, dt \quad \text{— (c)}$$

where the integrand $\mathcal{L}$ is called the Lagrangian.

⇒ A typical choice of $\mathcal{L}$ is "effort", modeled as a quadratic function of the control.

$$\mathcal{L} = u^T W u$$

{ where W is positive definite weighting matrix }

⇒ Another common choice is to leave t_f free and choose $\mathcal{L} = 1$, implying $J = t_f$.

↳ minimum-time problem

⇒ For now, let us ignore the issue of actuator limits & obstacles, and assume that the final time t_f is fixed.

⇒ The problem then is to find a state & control history

$$(x(t), u(t)), \quad t \in [0, t_f]$$

that satisfies the constraints of eq (b), satisfies the terminal condition

$$x(t_f) = x_f$$

and minimize the cost in eq ⓒ.

⇒ To write a necessary condition for optimality, we define the Hamiltonian H.

$$H(x(t), u(t), \lambda(t), t) = \mathcal{L}(x(t), u(t), t) + \lambda^T f(x(t), u(t))$$

where $\lambda(t)$ is a vector of Lagrange multipliers.

⇒ Then the Pontryagin minimum principle says that at an optimal solution $(x^*(t), u^*(t), \lambda^*(t))$

$$H^* = H(x^*(t), u^*(t), \lambda^*(t), t) \leq H(x^*(t), u(t), \lambda^*(t), t)$$

{ If the control history $u^*(t)$ is optimal, then at any time t, any other feasible control $u(t)$ will give an $H(t)$ greater than or equal to that of the optimal $H^*(t)$ }

⇒ In the absence of another Constraints on the state & control, then a necessary & sufficient Condition for optimality can be written

$$\frac{\delta H}{\delta u} = 0 \quad \& \quad \frac{\delta^2 H}{\delta u^2} > 0$$

⇒ This is known as the convexity or Legendre-Clebsch condition.

⇒ The Lagrange multipliers evolve according to the adjoint equation.

$$\dot{\lambda} = -\frac{\delta H}{\delta x}$$

⇒ What if there are actuator limits or obstacles.

- → At an optimal solution, the minimum Principle will always be satisfied, but the control or state history may bump up against limits.
- → Optimal Solution may be Constrained by these limits rather that the extremality condition.
- → Optimal Solution for this problem is a bang-bang trajectory, just like the bang-bang trajectory found by the time-scaling algorithm.

⇒ Only for very simple systems it is possible to Solve the extremality condition analytically.

- → In most cases it is necessary to resort to numerical methods to solve the condition approximately.
  - → dynamic programming
  - → gradient based nonlinear optimization.

★ Grid-Based Search

⇒ An alternative approach, specifically for time optimal trajectory planning, uses grid search.

⇒ As motivation, consider the simple double-integrator system

$$\ddot{q} = a$$
$$|a| \leq a_{max}$$

{where a is the acceleration control}

⇒ Let q be one-dimensional.

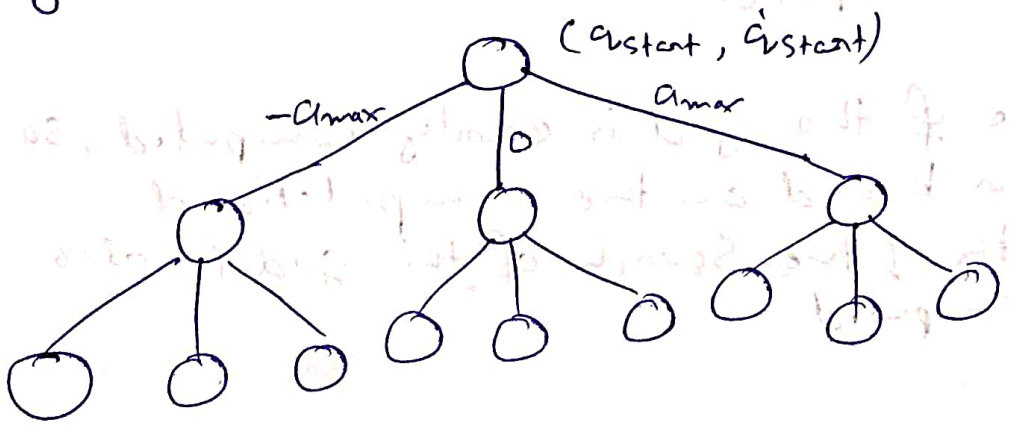
⇒ The time-optimal control from an initial state $(q_{start}, \dot{q}_{start})$ to a goal state $(q_{goal}, \dot{q}_{goal})$ is bang-bang:

↳ Actuator is saturated all the times.

⇒ First, we discretize the control set to $\{-a_{max}, 0, a_{max}\}$

⇒ Next, we choose a timestep h.

⇒ Now, beginning from $(q_{start}, \dot{q}_{start})$, we integrate the three controls forward in time by h to obtain three new states.



⇒ We continue in a breadth-first fashion.

⇒ If the trajectory to a new node in the tree passes through an obstacle or exceeds a velocity limit, this node is pruned from the tree.

⇒ The search continues until a trajectory reaches a state in a specified goal region.

⇒ The trajectory is specified by the piecewise-constant controls to traverse the tree from the root node to this final node.

⇒ Since the search is breadth-first exploring all reachable states at time Kh before moving on to time $(K+1)h$.

↳ The trajectory is time-optimal for the chosen discretization of time & controls.

⇒ We call this a grid-based search because each of the nodes reached during the growth of the tree lies on a regular grid on the $(q, \dot{q})$ state space.

↳ The size of the grid is easily computed, so an upper bound on the computational complexity of the search of this grid is also easily computed.

⇒ Let us now Consider a more general problem statement.

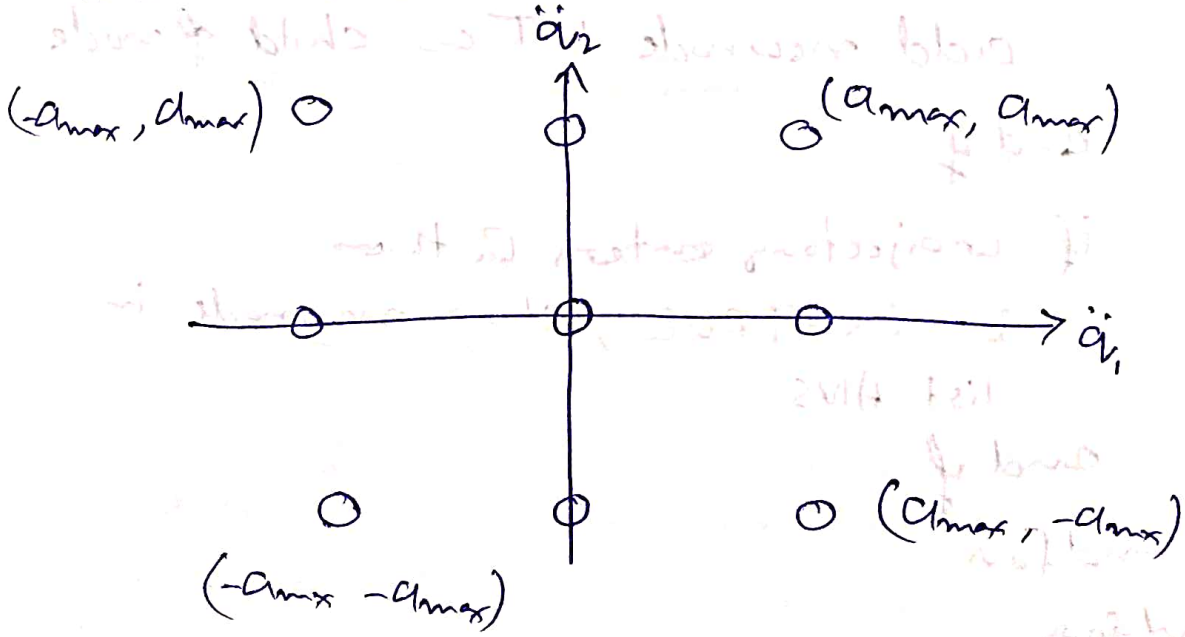
⇒ The system is described by $q \in D \subset R^{n_Q}$, where D is a bounded subset of R^{n_Q}, and velocity & acceleration bounds of the form

$$|\dot{q}_i| \leq v_{max}, \quad i = 1, \ldots n_Q$$

$$|\ddot{q}_i| < a_{max}, \quad i = 1, \ldots n_Q$$

⇒ The problem is to find a collision-free, approximately time optimal trajectory from $(q^*_{start}, \dot{q}^*_{start})$ to a goal state near the desired goal state $(q_{goal}, \dot{q}_{goal})$

⇒ The algorithm uses a discretized control set A Consisting of the cross products of $\{-a_{max}, 0, a_{max}\}$ for each degree of freedom, yielding 3^{n_Q} distinct control.



Algorithm: GRID_SEARCH

Input: Start node $(q^*_{start}\ \dot{q}^*_{start})$, Goal region G

Output: A trajectory to G or FAILURE

---

1. Place $(q^*_{start}, \dot{q}^*_{start})$ at root of tree T (level 0)
2. level ← 0, solved ← FALSE, ANS ← ∅
3. While not solved do
4. if no nodes in level level of T then
5. return FAILURE
6. end if
7. for each node in level of T do
8. for each control in A do
9. Integrate control for time h from node, getting newnode.
10. if newnode has not been previously reached, and trajectory does not pass close to an obstacle nor exceed Vmax then
11. add newnode to T as child of node
12. end if
13. if trajectory enters G then
14. Solved ← TRUE, store newnode in list ANS
15. end if
16. endfor
17. endfor

18. level ← level + 1
19. end while
20. For each node in ANs, find the trajectory that reached G first