Adatto a tesi di laurea triennale o a progetto per il corso di Ottimizzazione combinatoria.

1 Grand Prix

Assume a set of points is given in a 2-dimensional or 3-dimensional space (for instance a subset of points on a grid). They represent positions that a moving object can go through (possibly within a tolerated approximation). The movement is subject to a certain inertia, which is assumed to depend on the speed of the object (assume the speed values are discretized). Therefore for each point and each speed value, only some movements are allowed to a next point, representing limits to the allowed acceleration and deceleration of the object. The objective is to compute an optimal trajectory from a (set of) starting point(s) to a (set of) final point(s).

A typical and simplified setting is the following. The space is 2-dimensional; the points lie on a grid; they are obtained by drawing two continuous closed curves, one containing the other; the selected points are those between the two curves. The object speed is decomposed into its two components (horizontal and vertical), which are assumed to be independent. Each of them can vary (increase or decrease) by at most 1 unit when the object moves from a point to the next one. The objective is to compute a complete tour around the curves from a selected point to itself with a minimum number of movements.

The method suggested is dynamic programming.

The state is represented by the tuple (label) $S = (i, v_x, v_y, n)$, where i is the point reached, v_x and v_y are the speed components of the last move, n is the number of moves made so far. A state $S' = (i', v'_x, v'_y, n')$ dominates a state $S'' = (i'', v''_x, v''_y, n'')$ if

$$\begin{cases} i' = i'' \\ v'_x = v''_x \\ v'_y = v''_y \\ n' < n'' \end{cases}$$

Each state can have at most 9 successors, owing to the rule above. The algorithm iteratively extends states from points that are completely labeled, i.e. that have received all labels from all their predecessors. In order to define the precedence relation between points, it is enough to compute a (partial) order of the points, according to the orientation of the tour to be computed. This should be done in a pre-processing step.

The search can be made bi-directional and enriched with bounding to early detect infeasible or sub-optimal states.

Reference: https://codegolf.stackexchange.com/questions/32622/to-vectory-the-vector-racing-grand-prix

It is interesting to study more sophisticated variants, to compute optimal trajectories in real cases.