Volume xx (*200y*), *Number z*, *pp. 1–13*

Multi-Agent Path Planning with Asymmetric Interactions In Tight Spaces Supplementary Material

V. Modi[1] Y. Chen [1], A. Madan [1], S. Sueda [2], and D. I. W. Levin [1]

[1]University of Toronto, Toronto, Canada
[2]Texas A&M University, College Station, TX

1. Controlling Agent Behavior

Controlling agent behavior is very intuitive in our method. There are several points of control in our simulation. First, each agent has its own pre-determined characteristics profile. This includes things such as size, mass (doubles as stubbornness), collision radius, grouping preferences, preferred arrival time. Each of these parameters can be intuitively adjusted for any agent to get the desired behavior. Another control point for this method is adjusting the coefficients of the energy. If the scene happens to be on a large, unconstrained map, its advantageous to turn $K^M = 0$ off to prevent possible numerical issues from the smooth distance function. Agents are allowed to collide if K^C is set to 0. The grouping cost can be modified through the grouping parameter for each agent pair K^G. If agents need to reach their goal by a specific end time, we use K^T to control this behavior. Acceleration is controlled by the K_A parameter. Regularizing the edge lengths of each curve is important, as explained above, but modifying K^R can affect the likelihood of the solver getting stuck in local minimums in highly constrained environments. Last, but not least, K^K weights the kinetic energy term and increasing it (or increasing the mass term m) will incentivize the agent to follow a more direct path to the end location. It is very intuitive to play with these parameters in order to get the desired behavior in your multi-agent simulation scene.

1.1. Cost Function Pseudo-code

2. Discrete Gradients

Included below are the discrete gradients for each of our cost terms. For the sake of brevity, we do not include the gradient assembly/aggregation steps, only the element-wise gradient terms.

2.1. Kinetic Gradient

The kinetic energy gradient below is aggregated over each trajectory curve segment $e^i = [x^i, y^i, t^i, x^{i+1}, y^{i+1}, t^{i+1}]$ for each agent in the entire scene.

$$\begin{aligned}
\nabla_{x^i} C^K(\mathbf{s}_a) &= -K_a^K m_a \frac{x^{i+1}-x^i}{t^{i+1}-t^i} && (1)\\
\nabla_{y^i} C^K(\mathbf{s}_a) &= -K_a^K m_a \frac{y^{i+1}-y^i}{t^{i+1}-t^i} && (2)\\
\nabla_{t^i} C^K(\mathbf{s}_a) &= \frac{1}{2} K_a^K m_a \frac{(x^{i+1}-x^i)^2+(y^{i+1}-y^i)^2}{(t^{i+1}-t^i)^2} && (3)\\
\nabla_{x^{i+1}} C^K(\mathbf{s}_a) &= -\nabla_{x^i} C^K(\mathbf{s}_a) && (4)\\
\nabla_{y^{i+1}} C^K(\mathbf{s}_a) &= -\nabla_{y^i} C^K(\mathbf{s}_a) && (5)\\
\nabla_{t^{i+1}} C^K(\mathbf{s}_a) &= -\nabla_{t^i} C^K(\mathbf{s}_a) && (6)\\
& && (7)
\end{aligned}$$

Algorithm 1 The cost function is a summation of all individual costs. This is used in the interior point solve.

```
Require: global input variables from main paper
Require: @ALPHAHEURISTIC (main paper)
Require: @MarkedAsCloseBy (manually denoted agents are close-by)
Require: @BVH (A 3D tree structure storing agent trajectory nodes)
Require: @Overlaps (broad-phase check if two BVHs overlap)
  function COSTS($\mathbf{S}, \alpha_0, K, R, T, M, b_v, T^{max}, \mu$)
      $f \leftarrow 0$
      for $a, b \varepsilon (0, .., \|A\|)$ do
          $\mathbf{s}_a, \mathbf{s}_b$ from $\mathbf{S}$
          $r_a, r_b$ from $R$
          $K^C, K^G$ from $K$
          $\alpha \leftarrow ALPHAHEURISTIC(\alpha_0, \mathbf{s}_a, \mathbf{s}_b)$
          if MarkedAsCloseBy(A,B) or Overlaps(BVH(A), BVH(B)) then
              $f \leftarrow f + \mu C^C(\mathrm{d}(\alpha, \mathbf{s}_a, \mathbf{s}_b), K^C, r^a, r^b)$
              $f \leftarrow f + \mu C^G(\mathrm{d}(\alpha, \mathbf{s}_a, \mathbf{s}_b), K^G, r^a, r^b)$
          end if
      end for
      for $a \varepsilon (0, .., \|A\|)$ do
          $\alpha \leftarrow ALPHAHEURISTIC(\alpha_0, \mathbf{s}_a, b_v)$
          $\mathbf{s}_a, r_a, m_a, T_a^p, T_a^{max}$ from $\mathbf{S}, R, M, T, T^{max}$
          $K^K, K^A, K^T, K^R, K^M$ from $K$
          $f \leftarrow f + C^M(\mathrm{d}(\alpha, \mathbf{s}_a, b_v), K^M, r^a)$
          $f \leftarrow f + C^K(\mathbf{s}_a, m_a, K^K) + C^A(\mathbf{s}_a, K^A)$
          $f \leftarrow f + C^T(\mathbf{s}_a, K^T, T_a^p) + C^R(\mathbf{s}_a, K^R)$
      end for
      return $f$
  end function
```

2.2. Acceleration Gradient

The accelation energy measures the bend between two rod segments e^{i-1} and e^i. The gradient is aggregated over the non-boundary nodes of each trajectory curve $\mathbf{n}^i = [x^i, y^i, t^i]$ where $i \neq 0, n$ for an agent in the scene. Using the chain rule, we get

$$\nabla_{\mathbf{n}^{i-1}} C^A(\mathbf{s}_a) = \frac{\partial C^A}{\partial \theta^i} \frac{\partial \theta^i}{\partial n^{i-1}} \tag{8}$$

$$\nabla_{\mathbf{n}^{i}} C^A(\mathbf{s}_a) = \frac{\partial C^A}{\partial \theta^i} \frac{\partial \theta^i}{\partial n^{i}} \tag{9}$$

$$\nabla_{\mathbf{n}^{i+1}} C^A(\mathbf{s}_a) = \frac{\partial C^A}{\partial \theta^i} \frac{\partial \theta^i}{\partial n^{i+1}} \tag{10}$$

After further chain rule and breaking down each partial into its individual components, we get

$$v^i = \mathbf{n}^{i+1} - \mathbf{n}^i \tag{11}$$

$$v^{i+1} = \mathbf{n}^i - \mathbf{n}^{i-1} \tag{12}$$

$$\nabla_{\mathbf{n}^{i-1}} C^K(\mathbf{s}_a) = -K^A \theta^i \left(\frac{1}{1 + (\frac{\|v^i \times v^{i+1}\|}{v^i \cdot v^{i+1}})} \right) \left(\frac{v^i \cdot v^{i+1} \frac{1}{\|v^i \times v^{i+1}\|} * \left((v^i \times v^{i+1})^T [v^{i+1}]_{\mathrm{M}} \right) - v^{i+1} \|v^i \times v^{i+1}\|}{(v^i \cdot v^{i+1})^2} \right) \tag{13}$$

$$\nabla_{\mathbf{n}^{i+1}} C^K(\mathbf{s}_a) = K^A \theta^i \left(\frac{1}{1 + (\frac{\|v^i \times v^{i+1}\|}{v^i \cdot v^{i+1}})} \right) \left(\frac{v^i \cdot v^{i+1} \frac{1}{\|v^i \times v^{i+1}\|} * \left((v^i \times v^{i+1})^T [v^{i}]_{\mathrm{M}} \right) - v^{i} \|v^i \times v^{i+1}\|}{(v^i \cdot v^{i+1})^2} \right) \tag{14}$$

$$\nabla_{\mathbf{n}^{i}} C^K(\mathbf{s}_a) = -\nabla_{\mathbf{n}^{i-1}} C^K(\mathbf{s}_a) - \nabla_{\mathbf{n}^{i+1}} C^K(\mathbf{s}_a) \tag{15}$$

In the acceleration gradient, we manually set $\nabla_{t^{i-1}} C^K(\mathbf{s}_a) = \nabla_{t^i} C^K(\mathbf{s}_a) = \nabla_{t^{i+1}} C^K(\mathbf{s}_a) = 0$ since the kinetic gradient already controls acceleration in the t direction.

2.3. Preferred End Time Gradient

This gradient only affects the t^n for an agent as shown below

$$\nabla_{t^n} C^T(\mathbf{s}_a) = K^T(t^n - T^p) \tag{16}$$

2.4. Regularizer Gradient

The regularizer gradient below is aggregated over each trajectory curve segment $e^i = [x^i, y^i, t^i, x^{i+1}, y^{i+1}, t^{i+1}]$ for each agent in the entire scene. The regularizer energy and by extension, gradient only regularize time intervals over the trajectory as follows:

$$\nabla_{t^i} C^R(\mathbf{s}_a) = K^T \frac{\frac{t^n}{n}}{(t^{i+1} - t^i)^2} \tag{17}$$

$$\nabla_{t^{i+1}} C^R(\mathbf{s}_a) = -K^T \frac{\frac{t^n}{n}}{(t^{i+1} - t^i)^2}. \tag{18}$$

2.5. Collision Interaction Gradient

The collision gradient for the interaction between agent a and agent b is aggregated over each curve segment for the two trajectories. First we take the derivatives of the LogSumExp smooth distance function $\mathrm{d}(\alpha, \mathrm{u}(\mathbf{s}_a), \mathrm{u}(\mathbf{s}_b))$ w.r.t to the upsampled trajectory curves of size $m >> n$ $\mathrm{u}(\mathbf{s}) = \mathrm{upsample}(\mathbf{s})$ with $k, q \varepsilon 0, .., m$ denoting the upsampled node indices.

$$\frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} = \frac{-1}{\alpha} e^{-\alpha \|\mathrm{u}_a^k - \mathrm{u}_b^q\|} \frac{\mathrm{u}_a^k}{\|\mathrm{u}_a^k - \mathrm{u}_b^q\|} \tag{19}$$

$$\frac{\partial \mathrm{d}}{\partial \mathrm{u}_b^q} = \frac{-1}{\alpha} e^{-\alpha \|\mathrm{u}_a^k - \mathrm{u}_b^q\|} \frac{-\mathrm{u}_b^q}{\|\mathrm{u}_a^k - \mathrm{u}_b^q\|} \tag{20}$$

Next, we put the LogSumExp derivatives together with the $m \times n$ jacobians $J_a = \frac{\partial \mathrm{u}_a}{\partial \mathbf{s}_a}$ and $J_b = \frac{\partial \mathrm{u}_b}{\partial \mathbf{s}_b}$ into the gradients

$$\nabla_{\mathbf{s}_a} C^C(\mathrm{d}(\alpha, \mathrm{u}(\mathbf{s}_a), \mathrm{u}(\mathbf{s}_b)), r_a, r_b) = -\frac{K_a^C}{-r_a - r_b + \mathrm{d}} \frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} J_a \tag{21}$$

$$\nabla_{\mathbf{s}_b} C^C(\mathrm{d}(\alpha, \mathbf{s}_a, \mathbf{s}_b), r_a, r_b) = -\frac{K_b^C}{-r_a - r_b + \mathrm{d}} \frac{\partial \mathrm{d}}{\partial \mathrm{u}_b^k} J_b \tag{22}$$

Summed up over each agent-agent interaction interaction in the scene, this gives us the collision gradient.

2.6. Grouping(Friendship) Interaction Gradient

We use the same notation and jacobians ($\frac{\partial u}{\partial \mathbf{s}}$) as the Collision Gradient above to describe the Grouping cost's gradient

$$\nabla_{\mathbf{s}_a} C^G(\mathrm{d}(\alpha, \mathbf{s}_a, \mathbf{s}_b), r_a, r_b) = 2K_a^G(-r_a - r_b + \mathrm{d}) \frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} J_a \tag{23}$$

$$\nabla_{\mathbf{s}_b} C^G(\mathrm{d}(\alpha, \mathbf{s}_a, \mathbf{s}_b), r_a, r_b) = 2K_b^G(-r_a - r_b + \mathrm{d}) \frac{\partial \mathrm{d}}{\partial \mathrm{u}_b^k} J_b \tag{24}$$

2.7. Map Interaction Gradient

For interactions with the map boundary vertices b_v, we calculate the gradients using similar notation as above, except since the boundary is 2D, we can zero out the t dimension before passing points into the LogSumExp distance function. The map gradient for agent a is

$$\nabla_{\mathbf{s}_a} C_a^M(\mathrm{d}(\alpha, \mathbf{s}_a, b_v), r_a,) = \frac{K_a^M}{(-r_a + \mathrm{d})} \frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} J_a \tag{25}$$

$$\tag{26}$$

2.8. Gradient Pseudocode

Algorithm 2 The gradient is assembled as a sump of the individual components. In the appendix we include the gradient formulations. The resulting gradient is used in the interior point (IPSover).

```
Require: global input variables from Algorithm in main paper
Require: @ALPHAHEURISTIC (main submission)
Require: @MarkedAsCloseBy (manually denoted agents are close-by)
Require: @BVH (A 3D tree structure storing agent trajectory nodes)
Require: @Overlaps (broad-phase check if two BVHs overlap)
  function GRADS($\mathbf{S}, \alpha_0, K, R, T, M, b_v, T^{max}, \mu$)
     $g \leftarrow \mathbf{0}$
     for $a, b \varepsilon (0, .., \|A\|)$ do
        $\mathbf{s}_a, \mathbf{s}_b$ from $\mathbf{S}$
        $r_a, r_b$ from $R$
        $K^C, K^G$ from $K$
        $\alpha \leftarrow ALPHAHEURISTIC(\alpha_0, \mathbf{s}_a, \mathbf{s}_b)$
        if MarkedAsCloseBy(A,B) or Overlaps(BVH(A), BVH(B)) then
           $g \leftarrow g + \mu \nabla C^C(\mathrm{d}(\alpha, \mathbf{s}_a, \mathbf{s}_b), K^C, r^a, r^b)$
           $g \leftarrow g + \mu \nabla C^G(\mathrm{d}(\alpha, \mathbf{s}_a, \mathbf{s}_b), K^G, r^a, r^b)$
        end if
     end for
     for $a \varepsilon (0, .., \|A\|)$ do
        $\alpha \leftarrow ALPHAHEURISTIC(\alpha_0, \mathbf{s}_a, b_v)$
        $\mathbf{s}_a, r_a, m_a, T_a^p, T_a^{max}$ from $\mathbf{S}, R, M, T, T^{max}$
        $K^K, K^A, K^T, K^R, K^M$ from $K$
        $g \leftarrow g + \nabla C^M(\mathrm{d}(\alpha, \mathbf{s}_a, b_v), K^M, r^a)$
        $g \leftarrow g + \nabla C^K(\mathbf{s}_a, m_a, K^K) + \nabla C^A(\mathbf{s}_a, K^A)$
        $g \leftarrow g + \nabla C^T(\mathbf{s}_a, K^T, T_a^p) + \nabla C^R(\mathbf{s}_a, K^R)$
     end for
     return g
  end function
```

3. Discrete Hessians and Approximations

3.1. Kinetic Hessian

The kinetic hessian below is aggregated over each trajectory curve segment $e^i = [x^i, y^i, t^i, x^{i+1}, y^{i+1}, t^{i+1}]$ for each agent $a = [1, \|A\|]$ in the entire scene. The element-wise block is

$$m_a \begin{bmatrix} \frac{1}{(t^{i+1}-t^i)} & 0 & 0 & -\frac{1}{(t^{i+1}-t^i)} & 0 & 0 \\ 0 & \frac{1}{(t^{i+1}-t^i)} & 0 & 0 & -\frac{1}{(t^{i+1}-t^i)} & 0 \\ 0 & 0 & \frac{(x^{i+1}-x^i)^2+(y^{i+1}-y^i)^2}{(t^{i+1}-t^i)^3} & 0 & 0 & -\frac{(x^{i+1}-x^i)^2+(y^{i+1}-y^i)^2}{(t^{i+1}-t^i)^3} \\ -\frac{1}{(t^{i+1}-t^i)} & 0 & 0 & \frac{1}{(t^{i+1}-t^i)} & 0 & 0 \\ 0 & -\frac{1}{(t^{i+1}-t^i)} & 0 & 0 & \frac{1}{(t^{i+1}-t^i)} & 0 \\ 0 & 0 & -\frac{(x^{i+1}-x^i)^2+(y^{i+1}-y^i)^2}{(t^{i+1}-t^i)^3} & 0 & 0 & \frac{(x^{i+1}-x^i)^2+(y^{i+1}-y^i)^2}{(t^{i+1}-t^i)^3} \end{bmatrix} \tag{27}$$

3.2. Acceleration Hessian

The acceleration hessian measures the second order effects of the bend between two rod segments e^{i-1} and e^{i} in the trajectory curve of agent a. It is aggregated over the inside nodes, $\mathbf{n}^i = [x^i, y^i, t^i]$ where $i \neq 0, n$, of the trajectory curve. WLOG, for the non-boundary node i, we subdivide the symmetric $9x9$ hessian into the following unique $3x3$ blocks

$$\frac{\partial^2 C_a^A}{\partial n^{i-1} \partial n^{i-1}} = H^{1,1} \tag{28}$$

$$\frac{\partial^2 C_a^A}{\partial n^{i-1} \partial n^{i+1}} = H^{1,3} \tag{29}$$

$$\frac{\partial^2 C_a^A}{\partial n^{i+1} \partial n^{i+1}} = H^{3,3}. \tag{30}$$

The indexes of the block correspond to their placement in the $9x9$ hessian. We put together the node-wise hessian as

$$\begin{bmatrix} H^{1,1} & -H^{1,1} - H^{1,3} & H^{1,3} \\ (-H^{1,1} - H^{1,3})^T & -(-H^{1,1} - H^{1,3})^T - (-(H^{1,3})^T - H^{3,3})^T & (-(H^{1,3})^T - H^{3,3})^T \\ (H^{1,3})^T & -(H^{1,3})^T - H^{3,3} & H^{3,3} \end{bmatrix}. \tag{31}$$

For brevity, we define the following terms for use in just the acceleration hessian section:

$$d = v^i \cdot v^{i+1} \tag{32}$$

$$n = v^i \times v^{i+1} \tag{33}$$

$$c = \|v^i \times v^{i+1}\| \tag{34}$$

$$q = \operatorname{atan2}(n, d) \tag{35}$$

$$m = \frac{1}{1 + \left(\frac{n}{d}\right)^2} \tag{36}$$

$$L = [v^i]_{\mathrm{M}} \text{ cross product matrix of } v^i \tag{37}$$

$$M = [v^{i+1}]_{\mathrm{M}} \text{ cross product matrix of } v^{i+1} \tag{38}$$

$$T = (M \otimes v^{i+1})_{\gamma} c^{\gamma} \text{ tensor contraction} \tag{39}$$

The three unique components of the hessian are

$$\begin{aligned} H^{1,1} = &-m\left(-m\frac{\frac{d}{n}(c)^T[v^{i+1}]_{\mathrm{M}} - v^{i+1}n}{d^2}\right)\left(\frac{\frac{d}{n}(c)^T[v^{i+1}]_{\mathrm{M}} - v^{i+1}n}{d^2}\right)^T \\ &+ -q\left(\left(\frac{2\frac{n}{d}}{m^2}\right)\frac{\frac{d}{n}(c)^T[v^{i+1}]_{\mathrm{M}} - v^{i+1}n}{d^2}\right)\left(\frac{\frac{d}{n}(c)^T[v^{i+1}]_{\mathrm{M}} - v^{i+1}n}{d^2}\right)^T \\ &+ -qm\left(\frac{d^2n((Mc(v^{i+1})^T - d(M)^TM) + (v^{i+1}2c^TM)) - (dc^TM - (v^{i+1})^Tn^2) * (-2dnv^{i+1} + -(d^2(\frac{1}{n})(c^TM)))^T}{(d^2n)^2}\right), \end{aligned} \tag{40}$$

$$H^{1,3} = -m\left(m\frac{\frac{d}{n}(c)^T[v^{i}]_{\mathrm{M}} - v^{i}n}{d^2}\right)\left(\frac{\frac{d}{n}(c)^T[v^{i+1}]_{\mathrm{M}} - v^{i+1}n}{d^2}\right)^T \tag{41}$$

$$+ -q\left(\left(-\frac{2\frac{n}{d}}{m^2}\right)\frac{\frac{d}{n}(c)^T[v^{i}]_{\mathrm{M}} - v^{i}n}{d^2}\right)\left(\frac{\frac{d}{n}(c)^T[v^{i+1}]_{\mathrm{M}} - v^{i+1}n}{d^2}\right)^T \tag{42}$$

$$+ -qm\left(\frac{d^2n(((M^Tc)(v^i)^T - d(M^TL^T + T)) + -(\mathrm{I}n^2 + 2v^{i+1}c^TL)) - (dc^TM - (v^{i+1})^Tn^2)((2dnv^i) + (d^2\frac{1}{n}(c^TL)))^T}{(d^2n)^2}\right), \tag{43}$$

$$H^{3,3} = m\left(m\frac{\frac{d}{n}(c)^T[v^{i}]_{\mathrm{M}} - v^{i}n}{d^2}\right)\left(\frac{\frac{d}{n}(c)^T[v^{i}]_{\mathrm{M}} - v^{i}n}{d^2}\right)^T \tag{44}$$

$$+q\left(\left(-\frac{2\frac{n}{d}}{m^2}\right)\frac{\frac{d}{n}(c)^T[v^{i}]_{\mathrm{M}} - v^{i}n}{d^2}\right)\left(\frac{\frac{d}{n}(c)^T[v^{i}]_{\mathrm{M}} - v^{i}n}{d^2}\right)^T \tag{45}$$

$$+qm\left(\frac{d^2n(((L^Tc)(v^i)^T-dL^TL^T)+-(v^i2c^TL))-(dc^TL-(v^i)^Tn^2)((2dnv^i)+(d^2\frac{1}{n}(c^TL)))^T}{(d^2n)^2}\right). \tag{46}$$

The resulting $9x9$ hessian is assembled over each interior node of each trajectory curve in the scene.

3.3. Preferred End Time Hessian

This is the simplest hessian. It is the constant matrix

$$\begin{bmatrix} 0 & & \\ & \ddots & 0 \\ & 0 & 1 \end{bmatrix} \tag{47}$$

for each agent in the scene since the only non-zero value in the hessian is $\frac{\partial^2 C_a^T}{\partial t^n \partial t^n}$

3.4. Regularizer Approximate Hessian

The regularizer hessian below is aggregated over each trajectory curve segment $e^i = [x^i, y^i, t^i, x^{i+1}, y^{i+1}, t^{i+1}]$ for each agent in the entire scene in a symmetric block-diagonal fashion. The approximation assumes the end time, t^n, for each agent is constant. Therefore, for each trajectory curve segment in the scene, we assemble to following block into the global hessian:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2K_a^R\frac{\frac{t^n}{n}}{(t^{i+1}-t^i)^3} & 0 & 0 & -2K_a^R\frac{\frac{t^n}{n}}{(t^{i+1}-t^i)^3} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2K_a^R\frac{\frac{t^n}{n}}{(t^{i+1}-t^i)^3} & 0 & 0 & 2K_a^R\frac{\frac{t^n}{n}}{(t^{i+1}-t^i)^3} \end{bmatrix} \tag{48}$$

3.5. Collision Interaction Approximate Hessian

The collision hessian approximation is an agent-wise block diagonal hessian with dense individual blocks. Each block is subsequently sparsified by the function in 3. The agent-wisedense approximation is the following

$$\nabla^2_{\mathbf{s}_a} C^C(\mathrm{d}(\alpha, \mathrm{u}(\mathbf{s}_a), \mathrm{u}(\mathbf{s}_b)), r_a, r_b) = \frac{K_a^C}{(-r_a - r_b + \mathrm{d})^2}\left(\frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} J_a\right)\left(\frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} J_a\right)^T. \tag{49}$$

This term is passed into the sparsification function which uses the kronecker product to zero out the indices of the hessian which correspond to vertices that have an absolute distance greater than the given cutoff value. The sparse hessian is subsequently assembled into the correct block of the overall hessian as shown in Figure 1. For simplicity, we drop the smoothing function's 2nd derivative from our hessian and drop

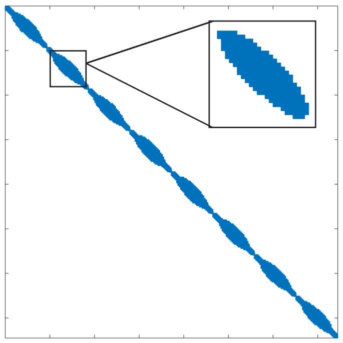

Figure 1: This is the assembled collision hessian matrix approximation where each block diagonal component has been passed through a sparsification function.

the off-diagonal terms. We find that dropping the off-diagonal blocks actually improves convergence and smoothness of trajectories.

3.6. Grouping(Friendship) Interaction Hessian

Using the same notation as in the collision interaction hessiani, we present the grouping hessian. This is passed through the same sparsification function as the collision hessian before assembly.

$$\nabla^2_{\mathbf{s}_a} C^G(\mathrm{d}(\alpha, \mathrm{u}(\mathbf{s}_a), \mathrm{u}(\mathbf{s}_b)), r_a, r_b) = 2K_a^G \left(\frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} J_a\right)\left(\frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} J_a\right)^T \tag{50}$$

3.7. Map Interaction Approximate Hessian

Again, we pass the following map interaction hessian through a sparsification function and then assemble it together into the global hessian.

$$\nabla^2_{\mathbf{s}_a} C^M(\mathrm{d}(\alpha, \mathrm{u}(\mathbf{s}_a), b_v), r_a, r_b) = \frac{K_a^M}{(-r_a + \mathrm{d})^2} \left(\frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} J_a\right)\left(\frac{\partial \mathrm{d}}{\partial \mathrm{u}_a^k} J_a\right)^T. \tag{51}$$

3.8. Hessian and Hessian Appx Pseudo-code

Algorithm 3 The hessian is assembled as a sump of the individual components. In the appendix we include the hessian formulations and approximations. Some of the components, H_C, H_M, are block diagonal, but have dense blocks. So we sparsify these blocks using the SPARSIFYHESS function. In the end, we have a sparse hessian to use in the interior point (IPSover).

```
Require: : global input variables from Algorithm in main paper
Require: @ALPHAHEURISTIC (main paper)
Require: @MarkedAsCloseBy (manually denoted agents are close-by)
Require: @BVH (A 3D tree structure storing agent trajectory nodes)
Require: @Overlaps (broad-phase check if two BVHs overlap)
  function HESS($\mathbf{S}, \alpha_0, K, R, T, M, b_v, T^{max}, \mu$)
      $H, H_M, H_G, H_C \leftarrow \mathbf{0}$
      for $a, b \varepsilon (0, .., ||A||)$ do
          $\mathbf{s}_a, \mathbf{s}_b$ from $\mathbf{S}$
          $r_a, r_b$ from $R$
          $K^C, K^G$ from $K$
          $\alpha \leftarrow ALPHAHEURISTIC(\alpha_0, \mathbf{s}_a, \mathbf{s}_b)$
          if MarkedAsCloseBy(A,B) or Overlaps(BVH(A), BVH(B)) then
              $H_C \leftarrow \mu \nabla^2 C^C(\mathrm{d}(\alpha, \mathbf{s}_a, \mathbf{s}_b), K^C, r^a, r^b)$
              $H_G \leftarrow \mu \nabla^2 C^G(\mathrm{d}(\alpha, \mathbf{s}_a, \mathbf{s}_b), K^G, r^a, r^b)$
          end if
          $H \leftarrow H + SPARSIFYHESS(H_C + H_G, \mathbf{s}_a, \mathbf{s}_b, cutoff)$
      end for
      for $a \varepsilon (0, .., ||A||)$ do
          $\alpha \leftarrow ALPHAHEURISTIC(\alpha_0, \mathbf{s}_a, b_v)$
          $\mathbf{s}_a, r_a, m_a, T_a^p, T_a^{max}$ from $\mathbf{S}, R, M, T, T^{max}$
          $K^K, K^A, K^T, K^R, K^M$ from $K$
          $H_M \leftarrow \nabla^2 C^M(\alpha, \mathbf{s}_a, b_v, K^M, r^a)$
          $H \leftarrow H + SPARSIFYHESS(H_M, \mathbf{s}_a, b_v, cutoff)$
          $H \leftarrow H + \nabla^2 C^K(\mathbf{s}_a, m_a, K^K) + \nabla^2 C^A(\mathbf{s}_a, K^A)$
          $H \leftarrow H + \nabla^2 C^R(\mathbf{s}_a, K^R)$
      end for
      return H
  end function

  function SPARSIFYHESS($H, V_1, V_2, cutoffDist$)
      $D \leftarrow maxAbsoluteDist(V_1, V_2)$
      for $i \varepsilon [0, rows(H)]$ do
          for $j \varepsilon [0, cols(H)]$ do
              if $H(i, j) < cutoffDist$ then
                  $H(i, j) \leftarrow 0$
              end if
          end for
      end for
  end function
```

4. Table of All Example Timings

This table shows our experiments of attempting to generate plausible initial paths using our space-time djikstras approach, the shortest path between the agent's start and end, using the shortest path with randomly jittered nodes to prevent overlaps. Ours is the only viable option that works for every example. Random jittering works when the number of agents and interactions is EXCEEDINGLY small. Naive shortest path works when there are no intersections between the agents or the map.

Scene	**Space-Time Djikstras**	**Shortest Path**	**Shortest Path with Random Jittering**
2 Circle	Y	N	Y
8 Circle	Y	N	Y
10 Circle	Y	N	N
20 Circle	Y	N	N
30 Circle	Y	N	N
Airplane Seats	Y	N	N
No Collision*	N/A	N/A	N/A
Symmetric	Y	N	N
Mass Based	Y	N	N
Size Based	Y	N	N
Size+Mass Based	Y	N	N
Grouping	Y	Y	Y
Roombas	Y	N	Y
Bottleneck	Y	N	N
Square Maze	Y	N	N
Circle Maze	Y	N	N
Subway Platform	Y	N	N
Safari	Y	N	N
Smily Face	Y	N	N
Battlefield	Y	N	N

Table 1: Generation of plausible initial paths with and without using Space-time Djikstras. Y - plausible path produced, N - not produced. *In this case, interactions are ignored. So any path is a feasible initial path.

5. Table of All Example Timings

Scene	Num Agents	DOFs	Layers	Djikstra Time(s) Per Agent	Avg Time(s) Per Iteration	Total Solve Time(s)
2 Circle	2	100	3	0.044	0.052	0.790
8 Circle	8	400	9	0.127	0.428	16.258
10 Circle	10	500	9	0.127	0.647	50.308
20 Circle	20	1000	9	0.140	2.316	254.330
30 Circle	30	1500	9	0.165	5.617	526.861
Airplane Seats	5	250	10	0.836	0.041	3.583
No Collision	3	90	5	0.131	0.012	0.456
Symmetric	3	90	5	0.134	0.028	2.355
Mass Based	3	90	5	0.134	0.050	3.820
Size Based	3	90	5	0.129	0.062	2.671
Size+Mass Based	3	90	5	0.131	0.012	0.346
Grouping	3	90	5	0.129	0.021	0.849
Roombas	3	105	10	0.131	0.015	2.049
Bottleneck	5	250	5	0.299	0.107	29.607
Square Maze	5	1000	3	6.343	1.095	83.240
Circle Maze	4	1200	3	1.175	2.877	385.526
Subway Platform	2	100	10	0.083	0.036	3.177
Safari	12	660	15	1.307	1.520	69.290
Smily Face	24	1080	20	4.000	4.497	535.118
Battlefield	102	6120	10	0.070	0.768	361.186

Table 2: Table of timings for all examples

6. All Agent Parameters

6.1. 2,8,10,20,30 Circles

- $\alpha_0 = 10$
- AgentTypes: Human
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 0.5, mass=1]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.5$
  - $MaxIts = 50, OuterIts = 5$

6.2. Airplane Seating

- $\alpha_0 = 200$
- AgentTypes: Human
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 10$, $K^A = 1$, $K^T = 0$, $K^R = 100$, $K^M = 1$, radius = 0.1, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 2.5
  - $\mu = 1, c = 0.75$
  - $MaxIts = 200, OuterIts = 1$

6.3. 3 Agent No Collisions

- $\alpha_0 = 30$
- AgentTypes: Human
- Human: [$K^C = 0$, $K^G = 0$, $K^K = 1$, $K^A = 1$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 0.5, mass = 1]
- Solver

- Djikstra's Agent Radius Multiplier = 1.5
- $\mu = 1, c = 0.5$
- $MaxIts = 40, OuterIts = 2$

6.4. 3 Agent Symmetric

- $\alpha_0 = 30$
- AgentTypes: Human
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 10$, $K^M = 0$, radius = 0.5, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.5$
  - $MaxIts = 40, OuterIts = 2$

6.5. 3 Agent Mass/Stubbornness Based

- $\alpha_0 = 30$
- AgentTypes: Human, Snake
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 0.5, mass = 1]
- Snake: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 0.5, mass = 10]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.5$
  - $MaxIts = 40, OuterIts = 5$

6.6. 3 Agent Size Based

- $\alpha_0 = 150$
- AgentTypes: Human, Deer
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 0.5, mass = 1]
- Deer: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 1.5, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.5$
  - $MaxIts = 40, OuterIts = 5$

6.7. 3 Agent Size+Mass Based

- $\alpha_0 = 150$
- AgentTypes: Human, Elephant
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 0.5, mass = 1]
- Elephant: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 1, mass = 10]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.5$
  - $MaxIts = 40, OuterIts = 5$

6.8. 3 Agent Grouping

- $\alpha_0 = 30$
- AgentTypes: Human
- Human: [$K^C = 1$, $K^G = 1$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 0.5, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.5$
  - $MaxIts = 40, OuterIts = 2$

6.9. Roombas

- $\alpha_0 = 100$
- AgentTypes: Roomba
- Roomba: [$K^C = 1$, $K^G = 0$, $K^K = 10$, $K^A = 1$, $K^T = 0$, $K^R = 100$, $K^M = 1$, radius = 0.1, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.5$
  - $MaxIts = 50, OuterIts = 2$

6.10. Bottleneck

- $\alpha_0 = 50$
- AgentTypes: Elephant
- Elephant: [$K^C = 1$, $K^G = 0$, $K^K = 10$, $K^A = 0$, $K^T = 10$, $K^R = 100$, $K^M = 1$, radius = 1, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.75$
  - $MaxIts = 100, OuterIts = 2$

6.11. Square Maze

- $\alpha_0 = 100$
- AgentTypes: Human
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 1$, radius = 0.5, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 2
  - $\mu = 1, c = 0.75$
  - $MaxIts = 30, OuterIts = 5$

6.12. Circle Maze

- $\alpha_0 = 100$
- AgentTypes: Human
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 1$, radius = 0.15, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.5$
  - $MaxIts = 30, OuterIts = 5$

6.13. Subway Platform

- $\alpha_0 = 150$
- AgentTypes: Human
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 10$, $K^M = 1$, radius = 0.35, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 2
  - $\mu = 1, c = 0.75$
  - $MaxIts = 150, OuterIts = 2$

6.14. Safari

- $\alpha_0 = 100$
- AgentTypes: Human, Baboon, Elephant
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 1$, $K^R = 10$, $K^M = 1$, radius = 0.25, mass = 1]
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 5$, $K^R = 10$, $K^M = 1$, radius = 0.25, mass = 1]

- Elephant: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 10$, $K^M = 1$, radius = 1.2, mass = 100]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.75$
  - $MaxIts = 40, OuterIts = 5$

6.15. Smily Face

- $\alpha_0 = 100$
- AgentTypes: Human
- Human: [$K^C = 1$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 0.25, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.8$
  - $MaxIts = 50, OuterIts = 5$

6.16. Battlefield

- $\alpha_0 = 40$
- AgentTypes: Human
- Human: [$K^C = 10$, $K^G = 0$, $K^K = 1$, $K^A = 0$, $K^T = 0$, $K^R = 100$, $K^M = 0$, radius = 0.25, mass = 1]
- Solver
  - Djikstra's Agent Radius Multiplier = 1.5
  - $\mu = 1, c = 0.8$
  - $MaxIts = 3, OuterIts = 100$