SC-M*: A Multi-Agent Path Planning Algorithm with Soft-Collision Constraint on Allocation of Common Resources

Rongye Shi [1,*] **, Peter Steenkiste** [1,2,*] **and Manuela M. Veloso** [3,*]

1   Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
2   Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
3   Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
*   Correspondence: rongyeshi@cmu.edu (R.S.); prs@cs.cmu.edu (P.S.); mmv@cs.cmu.edu (M.M.V.)

Featured Application: SC-M* generalizes the M* algorithm to address real-world multi-agent path planning problems in the soft-collision context, which considers the allocation of common resources requested by agents. Application examples include but are not limited to city-scale passenger routing in mass transit systems, network traffic engineering and planning for large-scale autonomous vehicles.

Abstract: Multi-agent path planning (MAPP) is increasingly being used to address resource allocation problems in highly dynamic, distributed environments that involve autonomous agents. Example domains include surveillance automation, traffic control and others. Most MAPP approaches assume hard collisions, e.g., agents cannot share resources, or co-exist at the same node or edge. This assumption unnecessarily restricts the solution space and does not apply to many real-world scenarios. To mitigate this limitation, this paper introduces a more general class of MAPP problems—MAPP in a soft-collision context. In soft-collision MAPP problems, agents can share resources or co-exist in the same location at the expense of reducing the quality of the solution. Hard constraints can still be modeled by imposing a very high cost for sharing. This paper motivates and defines the soft-collision MAPP problem, and generalizes the widely-used M* MAPP algorithm to support the concept of soft-collisions. Soft-collision M* (SC-M*) extends M* by changing the definition of a collision, so paths with collisions that have a quality penalty below a given threshold are acceptable. For each candidate path, SC-M* keeps track of the reduction in satisfaction level of each agent using a collision score, and it places agents whose collision scores exceed its threshold into a soft-collision set for reducing the score. Our evaluation shows that SC-M* is more flexible and more scalable than M*. It can also handle complex environments that include agents requesting different types of resources. Furthermore, we show the benefits of SC-M* compared with several baseline algorithms in terms of path cost, success rate and run time.

Keywords: multi-agent systems; planning; M* algorithm; shortest path finding; collision-free constraint; optimality and completeness

1. Introduction

Multi-agent path planning (MAPP) involves finding the set of least-cost paths for a set of agents co-existing in a given *graph* such that each of the agents is free from collision, where a collision is defined as at least two agents moving to the same location at the same time. MAPP attracts increasing attention due to its practical applications in multi-robot systems for surveillance automation, video gaming,

traffic control, and many other domains [1–4]. This problem is, however, difficult to solve because the configuration space grows exponentially with the number of agents in the system, incurring extremely heavy computational efforts. It is an NP-hard problem to find optimal solutions for MAPP in its general form [5].

Approaches to solving MAPP problems fold into three main categories: *coupled, decoupled* and *intermediate* [6]. *Coupled* approaches search the *joint configuration space* of the multi-agent system, which is the *Tensor product* of the free configuration spaces of all the individual agents. A popular coupled planner is the A* algorithm [7] that directly searches the whole joint configuration space, making such an approach computationally infeasible when the number of agents is large. Enhanced variants of A*, such as operator decomposition (OD), enhanced partial expansion A* (EPEA*), and iterative deepening A* (IDA*), can—to some extent—mitigate the exponential growth in the number of neighbors by improving the admissible heuristics [8–11]. Coupled approaches are optimal and complete, but usually at high computational cost. *Decoupled* approaches plan for each agent separately and then adjust the path to avoid collisions. Algorithms in this category are generally faster because they perform a graph search and collision-avoidance adjustment in low-dimensional spaces. However, optimality and completeness are not guaranteed [3,12].

Intermediate approaches lie between coupled and decoupled ones because they dynamically couple agents and grow the search space during the planning. In this way, the search space is initially small and grows when necessary. A few intermediate MAPP algorithms can guarantee optimality and completeness. State-of-the-art examples include Conflict-Based Search (CBS) [6,13]. CBS is a two-level algorithm. At the high level, conflicts are added into a *conflict tree* (CT). At the low level, solutions consistent with the constraints given by the CT are found and updated to agents. CBS behaves poorly when a set of agents is strongly coupled. Meta-agent CBS (MA-CBS) is then proposed by merging strongly coupled agents into a meta-agent to handle the strongly coupled scenarios.

The M* algorithm is a state-of-the-art coupled approach. It starts with decoupled planning and applies a strategy called *sub-dimensional expansion* to dynamically increase the dimensionality of the search space in regions in which agent collisions occur. In this way, an efficient graph search with a strict collision-free constraint can be achieved, while minimizing the explored portion of the joint configuration space. M* identifies which subsets of agents can be safely decoupled and hence plans for multi-agents in a lower-dimensional space. Compared to CBS and its variant MA-CBS, M* and its variants, e.g., recursive M* (rM*), have much more fine-grained control over some technical details, such as the management of conflict sets for better scalability. The fine-grained nature of M* allows it to be integrated into MA-CBS to take advantage of both [14]. Recent work extended both M* and CBS algorithms to handle the imperfect path execution due to unmodeled environments and delays [15,16].

Most fundamental MAPP approaches assume *hard collisions*, which means that solutions in which agents share resources (nodes or edges) are rejected. In many real world scenarios, some degree of resource sharing between agents is acceptable, so the hard-collision constraint needlessly over-constrains the solution space. This paper relaxes the hard collisions constraint by allowing some sharing of resources, including space and various services on edges/nodes, by agents. Such sharing reduces the quality of the path, i.e., the satisfaction level of the agent using it, but as long as the quality reduction for each path is below a settable threshold, the solution is acceptable. We call this concept *soft collisions*. Hard collisions are still supported by having a very strict threshold, i.e., a penalty for sharing is very high. The reduction in satisfaction level experienced by an agent caused by soft collisions on resources in its path is quantified using a *collision score*. In this paper, we develop a generalized version of the M* algorithm, called *soft-collision M* (SC-M*), for solving the MAPP problem in the soft-collision context. Note that we that we are not simply replacing hard with soft collisions, but instead introducing soft collisions as a generalization that allows modeling different types of collisions.

SC-M* extends M* by taking the perspective of soft collision on common resources. Specifically, SC-M* tracks the collision score of each agent and places agents whose collision scores exceed certain thresholds into a *soft-collision set* for *sub-dimensional expansion*, a technique that limits the search space

while maintaining the optimality of the algorithm with respect to the objective. In this way, SC-M* achieves improved scalability to handle a larger number of agents while limiting the probability of collisions on resources to a bound.

In this paper, we show that SC-M* has advanced flexibility and scalability for efficiently solving the MAPP problem in the soft-collision context where common resources are considered, and can handle complex environments (e.g., with multiple types of agents requesting multiple types of resources). We theoretically prove that SC-M* is *complete* and *suboptimal* under the soft-collision constraints on resources. Experimental results demonstrate the advantages/trade-offs of SC-M* in terms of path cost, success rate and run time against baseline SC-based MAPP planners, such as SC-A* and SC-CBS.

The rest of the paper is organized as follows. Section 2 discusses the motivation of soft collisions. Section 3 gives technical briefing of the M* algorithm. Section 4 presents our proposed SC-M* approach. Section 5 evaluates SC-M* in a grid public transit network. Finally, Section 6 concludes our work.

2. Motivation

In some planning problems, solutions in which agents share resources, i.e., they collide using the traditional MAPP problem definition, may be acceptable, at the cost of having a reduced level of agent satisfaction. Problems of this type have two properties in common: (1) Agents' satisfaction conditions are reduced when meeting at the same place; and (2) the extent of reduction in satisfaction depends on how long the dissatisfying situation lasts in terms of distance or time.

One motivating example of this type of problems involves mass transit systems, in which passengers have various preferences, even necessities, in terms of *common resources*, such as seat availability (necessary for seniors) and on-vehicle Wi-Fi supply (preferred by video viewers and game players during the trip). Passengers may interfere with one another on common resources in crowded situations. Individually optimal paths can cause serious interference, leading to low-quality experiences. Interference between passengers is soft because it is possible that they do not call for the same resource when they are on the same public vehicle. In addition, even when they call for the same resource and interfere, they are able to tolerate each other over a short time and distance. Intuitively, how likely a collision (intolerable interference) actually happens depends on: (1) whether the resource supply is less than the demands; and (2) how long the lack-of-supply condition lasts in terms of the time and distance that the passengers stay together. Passengers can be viewed as agents, moving through the transportation network. When planners plan for all the agents, sticking to eliminating any hard collision is neither necessary nor feasible. Thus, people are more interested in another problem: How can the resource received by all agents be maximized such that the probability of collision of each agent is less than a bound? This is an important topic of passenger-centered research [17–19].

In addition to public transit scenarios, other examples include: network traffic engineering, where multiple data streams can route through a router. Long streams will have a higher chance of being blocked when unexpected traffic spikes pop up, exceeding the link capacity [20]. How to maximize the throughput with a bounded chance of blocking is of great interest to researchers in the field of communications and computer networks.

Another example is planning for large-scale self-driving cars, where multiple cars can share the same lane, and the number of cars on a road will influence the chance of crashes among autonomous vehicles [21,22]. Scholars and engineers dealing with the fundamentals of autonomous vehicles in unstructured and dynamic environments aim to increase the road traffic while bounding the crash risk.

Military transportation also has the soft-collision property, in which transport aircrafts or vehicles are subject to higher risks to be detected and attacked by enemy troops when many of they move together due to path overlap for a long distance. Formally, as the transportation volume on a road increases, the degree of concealment decreases [23]. The dispatcher must bound the security risk when attempting to maximize the military transportation efficiency.

To support these application classes, we introduce the soft-collision property (related to common resources) to MAPP. SC-M*, introduced in this paper, is the first attempt to generalize M* to handle

real MAPP problems in a soft-collision context, considering various common resources requested by agents. Specifically, SC-M* changes M*'s definition of a collision so it can represent soft collisions on resources and their impact on an agent's dissatisfaction level. We show the advantages of the SC-M* against other SC-based MAPP solvers.

3. Technical Briefing of M*

Before introducing the SC-M* method, this section reviews the traditional MAPP problem and the M* algorithm [6].

3.1. MAPP Problem Definition

In this problem, we have m agents indexed by the set $I = \{1, \ldots, m\}$. Let the free configuration space of agent j be represented by the directed graph $G^j = \{V^j, E^j\}$. For any agent j, graph G^j is the same. The joint configuration space, which describes the set of all possible states of the multi-agent system, is defined as the *tensor product* of the graphs of all individual agents: $G = G^1 \times \cdots \times G^m$. G consists of a *joint vertex* set V and a *joint edge* set E. As an example, in a 2-D joint configuration space given by the agents j and k, the two 2-D joint vertexes $v_p = (v_p^j, v_p^k)$ and $v_q = (v_q^j, v_q^k)$ is connected by the joint edge (e_{pq}^j, e_{pq}^k). Note that $v_p^j \in V^j$ and $e_{pq}^j \in E^j$. Let $\pi^j(v_p^j, v_q^j)$ denote a sequence of joint vertexes, called a *path* in G^j from v_p^j to v_q^j. The cost of a path $\pi(v_p, v_q)$ in G is defined as

$$g(\pi(v_p, v_q)) = \sum_{j=1}^{m} g(\pi^j(v_p^j, v_q^j)), \tag{1}$$

where $g(\pi)$ is the sum of all edge costs involved in the joint path π.

The goal of MAPP is to find a collision-free path, which is optimal with respect to minimal cost, from the source configuration $v_s = v_s^1 \times \cdots \times v_s^m$ to the destination configuration $v_d = v_d^1 \times \cdots \times v_d^m$. To determine the collision between agents, a collision function $\psi(v_p)$ is defined to return the set of conflicting agents at v_p.

Most fundamental MAPP approaches use hard collisions, where no intersection is allowed between any two agents in terms of the occupation of any *resource*, such as a workspace. This implies that the capacity of each resource can support only one agent at a time (i.e., a collision happens immediately once agents intersect at any resource). Suppose we have a set of resources $A = \{A_1, \ldots, A_L\}$ requested by each agent in the multi-agent system, where A_i is defined as the set of resource of type i on all edges and vertexes in G. A_i is a continuous set because only continuous resources are considered in the paper. A traditional hard-collision constrained MAPP problem is formulated as follows:

$$\min_{\pi} g\left(\pi\left(v_s, v_d\right)\right)$$

$$s.t. \tag{2}$$

$$\bigcup_{\forall i \neq j \in I} \left(A_k(v_p^i) \cap A_k(v_p^j) \right) = \varnothing, \ \forall A_k \in A, \ \forall v_p \in \pi,$$

where $A_k(v_p^j)$ denotes the subset of resource A_k occupied by the agent j at the joint vertex v_p. One state-of-the-art solver to this problem is M*, which uses the sub-dimensional expansion strategy to dynamically increase the dimensionality of the search space in regions featuring some agent collisions. M* enables a relatively cheaper graph search under the strict hard-collision constraint.

3.2. Graphic-Centric Description of M*

This section uses the graphic-centric description introduced by wanger [6] to illustrate M*. M* is a complete and optimal MAPP algorithm. The main idea of M* is to iteratively construct/update a

so-called *search graph* G^{sch} (i.e., to iteratively remove the collision configuration vertexes and expand necessary neighbors) and apply the A* algorithm on the new G^{sch} until the optimal collision-free path to v_d exists in the G^{sch} and is found by the A* search. Specifically, G^{sch} is a sub-graph of G and consists of three other sub-graphs: the *expanded graph* G^{exp}, *neighbor graph* G^{nbh}, and *policy graph* G^ϕ. The expanded graph G^{exp} is the sub-graph of G that has been explored by M*. G^{nbh} contains the *limited neighbors* of all the joint vertexes in G^{exp}. The definition of limited neighbors is given below. G^ϕ consists of the paths induced by the *individually optimal policy* ϕ that connects each joint vertex in $G^{nbh} \cup G^{exp}$ to v_d without the collision-free constraint. Specifically, ϕ^j is the individually optimal policy for the agent j that leads any v^j in $G^{nbh} \cup G^{exp}$ to v_d^j without considering collisions. Examples of policy ϕ include the standard *Dijkstra's* algorithm [24] and A* [5]. Using the above graphic concepts, we can define the *collision set* C_p as

$$
C_p = \begin{cases} \psi\,(v_p) \cup \{\bigcup_{v_q \in \boldsymbol{V_p}} \psi\,(v_q)\}, & \text{for } v_p \in G^{exp} \\ \varnothing & \text{, for } v_p \notin G^{exp} \end{cases}, \tag{3}
$$

where $\boldsymbol{V_p} = \{v_q | \exists \pi(v_p, v_q) \subseteq G^{exp}\}$ is the set of the joint vertexes to which there exists a path to from v_p in G^{exp}. Let $\phi^j(v^j)$ be the immediate *successor vertex* of v^j in the policy path, then the set of *limited neighbors* V_p^{nbh} for the joint vertex v_p in G^{nbh} is defined as

$$
V_p^{nbh} = \left\{ v_q \,\middle|\, \begin{cases} e_{pq}^j \in E^j, & \text{for } j \in C_p \\ v_q^j = \phi^j\left(v_p^j\right), & \text{for } j \notin C_p \end{cases} \right\}, \tag{4}
$$

where $e_{pq}^j = edge\left(v_p^j, v_q^j\right)$. The definition of the limited neighbors implies the sub-dimensional expansion strategy: We only expand the search space at the dimensions where the collision occurs ($j \in C_p$), otherwise for collision-free dimensions ($j \notin C_p$), M* will not expand, limiting the unexpanded search space to the graph that only consists of individually optimal path induced by the policy ϕ.

3.3. Algorithm Description of M*

The high-level description of M* is as follows [6]: Initially, M* computes the individually optimal policy ϕ for each agent from source v_s to destination v_d. The initial search graph G^{sch} only consists of an individually optimal path: Initial G^{exp} contains v_s only; initial G^{nbh} contains $\phi(v_s)$ only, which is the successor of v_s along the individually optimal policy; and initial G^ϕ contains the optimal policy path from the vertex in G^{nbh} and G^{exp} all the way to v_d. $C_p = \varnothing$ for all v_p in initial G^{sch}. Given the initial G^{sch}, the A* algorithm is applied using the following *admissible heuristic*

$$
h\,(v_p) = g(\pi_\phi(v_p, v_d)\,) \le g(\pi_*(v_p, v_d)), \tag{5}
$$

where π_ϕ is the individually optimal path induced by policy ϕ, and π_* is the ground-truth optimal multi-agent path we want to find. The initial *open list* (i.e., *priority queue*) contains v_s only, with zero cost. The open list is sorted according to $v_p.cost + h\,(v_p)$, where $v_p.cost$ is the current cost of v_p from the source.

In each iteration, M* expands the first-ranked vertex v_p from the open list to G^{exp} and investigates each joint vertex v_q in the limited neighbors of v_p (i.e., $v_q \in V_p^{nbh}$) if no collision occurs at v_p; otherwise, it jumps to the next iteration. If there exists a collision (i.e., $\psi\,(v_q) \ne \varnothing$), M* will update the collision set C_q with $C_q \cup \psi\,(v_q)$, and this update will back-propagate from v_q to: (1) its immediate predecessor v_p; and (2) all the way back to any ancestors that have at least one path inside of G^{exp} leading to v_q (see Equation (3) for details). After this pre-processing, the algorithm:

- investigates and updates the cost of the vertex v_q and records its corresponding predecessor; and
- adds v_q and all its predecessors/ancestors, of which the collision sets are changed, to the open list.

This process is repeated until v_d is expanded or open list is empty.

The critical point is that: Only when a collision set C_p is changed will the search graph G^{sch} change. It is the operation of updating the collision set in a back-propagation way that makes the story different: By including $\psi(v_p)$ to C_p, M* can tell which agents are *immediately* collided at the current v_p; by including all $\psi(v_q)$ for $v_q \in V_p$ to C_p (i.e., the collision information of all the expanded downstream successors from v_p), M* can preview which agents will collide in the future, making it possible to pre-plan to avoid that. Therefore, using the limited neighbor set in Equation (4) makes sense: It advises M* to only expand the dimensions where there exists an immediate collision at v_p or there will be collisions in the future, starting from v_p, in the current expanded graph G^{exp}. Figure 1 shows an example of how M* solves the optimal collision-free path planning for the two agents.
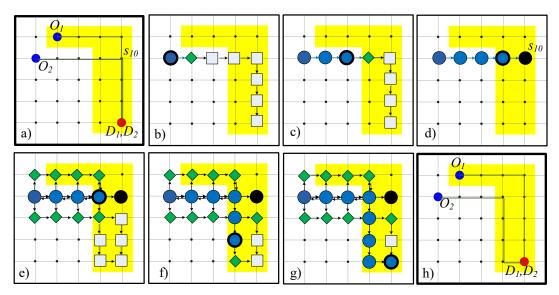

Figure 1. Illustration of traditional M* for two agents, where we show the evolution of the expanded graph G^{exp} (circle), neighbor graph G^{nbh} (diamond), and policy graph G^{ϕ} (square) for Agent 2 as the M* algorithm proceeds. (**a**) Individually optimal paths; (**b**) the first expanded vertex; (**c**) the third expanded vertex; (**d**) collision occurs at vertex s_{10}; (**e**) sub-dimensional expansion; (**f**) search in the expanded space; (**g**) the destination of Agent 2 founded; (**h**) collision-free optimal paths for both agents founded by M*.

In Figure 1, we can visualize the evolution of the search graph G^{sch} of Agent 2. G^{sch} consists of an expanded graph G^{exp} (circle), a neighbor graph G^{nbh} (diamond), and a policy graph G^{ϕ} (square). Edge cost and direction-changing cost are considered during planning. Yellow zones are preferred areas with lower edge cost. In M*, individually optimal paths are induced by ϕ for each individual agent (Figure 1a). We can observe that there will be a collision at vertex s_{10}, which is ignored by ϕ. For Agent 2, M* searches in the subspace, and the most promising vertex is expanded at each iteration (Figure 1b,c). Then, a collision occurs at vertex s_{10} and triggers the removal of the rest of G^{sch} (Figure 1d), which is equivalent to jumping to the next iteration. Following the sub-dimensional expansion strategy, M* extends the search space to include the limited neighbors, and a new G^{sch} is obtained (Figure 1e). By searching in the new G^{sch}, M* finds the optimal collision-free path for Agent 2 (Figure 1f,g). On the other hand, the planning for Agent 1 is conducted simultaneously, and, finally, the collision-free optimal paths for both agents are found by M* (Figure 1h).

4. Soft-Collision M* (SC-M*)

M* assumes hard-collision constraint which does not apply to many real-world applications. Our contribution in this paper is to generalize M* to soft-collision context where common resources are considered, and to introduce soft collisions as a generalized concept allows us to model different types

of collisions. In addition, we show the advantages and trade-offs of the proposed algorithm in this new scenario. The proposed SC-M* extends M* by changing the definition of a collision, so paths with hard collisions but with a level of dissatisfaction on resources below a given threshold are acceptable. In this section, we formulate the concept of soft collision on common resources, describe the generalized M* (i.e., soft-collision M*) for planning in the soft-collision context, and extend our approach to a more complex environment with multiple types of agents requesting multiple types of resources.

4.1. Soft-Collision Constraint on Common Resources

Inspired by real-world scenarios, we introduce the recourse-related soft-collision property to the model of an agent. We define that all the agents have the following properties: (1) a collision among agents is *soft*, quantified using some *collision scores*; and (2) different agents have different collision scores, according to their individual experiences through the paths. We suppose that each agent cares about a set of resources $A = \{A_1, \ldots, A_L\}$. To obtain the properties, we introduce to each agent an additional attribute called *resource experience* (for each resource) and use the resource experience to calculate the *collision score*.

In doing so, this section first uses the *resource experience* (as defined in Section 4.1.1 Definition 1) to quantify how dissatisfying the agent is about the resource allocated to it. Then, we combine this information of all the resources into a *collision score* (as defined in Section 4.1.2 Definition 2) that indicates the probability of the agent announcing a collision given its resource experience. *Threshold of collision* is used to limit the collision score, implying to what degree of unpleasantness we want to pursue the solution. The agent, of which the collision score exceeds the threshold, will be placed into a soft-collision set via the *soft-collision function* for sub-dimensional expansion (as defined in Section 4.1.3 Definition 3).

4.1.1. Definition 1 (Resource Experience)

We define *resource experience* to quantify the dissatisfying experience per resource about which an agent cares.

Let

- $\pi = \pi\,(v_s, v_b)$ be a path from the source v_s to some v_b;
- $v_q = \pi\,(v_p)$ be the immediate successor of v_p along the path π;
- $A_k(e^j_{pq})$ be the capacity (amount) of the subset of the resource A_k on the edge e^j_{pq}, given by the graph model; and
- $A^j_k(e^j_{pq})$ be the amount of the subset of the resource A_k actually *allocated* to the agent j on the edge e^j_{pq}, called the *allocated resource value*.

The *resource experience* is then defined as the *dissatisfying* experience of agent j on resource A_k along the path π^j:

$$D\left(\pi^j, A_k\right) = \sum_{v_p | v_p \in \pi_{/v_b}} \mathbf{1}\left(A_k(e^j_{pq}) \geq \varepsilon_k \wedge A^j_k(e^j_{pq}) < \varepsilon_k\right) \cdot g(e^j_{pq}), \tag{6}$$

where $\mathbf{1}(\cdot)$ is the indicator function, whose value is one if the logical condition is true, else zero; $\varepsilon_k \in \varepsilon = \{\varepsilon_1, \ldots, \varepsilon_L\}$ is the *satisfying value* regarding the resource A_k, which is a positive real value; $g(e^j_{pq})$ is the edge cost regarding travel time/distance given by the graph model; and $A^j_k(e^j_{pq})$ is formulated as:

$$A^j_k(e^j_{pq}) = \frac{A_k(e^j_{pq})}{\sum_{k \in I} \mathbf{1}\left(e^k_{pq} = e^j_{pq}\right)}. \tag{7}$$

Obviously, $A^j_k(e^j_{pq}) = A_k(e^j_{pq})$ if and only if no other agents are physically moving along with agent j on the edge e^j_{pq}. The allocated resource value $A^j_k(e^j_{pq})$ quantifies the level of interference

incurred by other agents when they physically move together. In contrast, the traditional hard-collision setting will always label a collision to the agent j and all other involved agents whenever $A_k^j(e_{pq}^j)$ is (even slightly) smaller than $A_k(e_{pq}^j)$. The resource experience is implemented as an attribute of the vertex class and can be calculated incrementally using Algorithm 1.

Algorithm 1 Function: experience(v_k, v_l, A).

Input: v_k: base vertex; v_l: immediate successor of the base vertex; A: list of resources
Output: v_l with updated experience

1: **for** A_p in A **do**

2:      **for** j in I **do**

3:          $v_l.\exp[A_p][j] \leftarrow v_k.\exp[A_p][j] + D\left(\pi(v_k, v_l)^j, A_p\right)$
4:      **end for**
5: **end for**
6: **return** v_l //the successor with updated experience

Combined with the allocated resource value, which serves as a proxy of the interference level, the definition of resource experience in Equation (6) actually defines a property of an agent: Only the situation in which the resource allocated to an agent is dissatisfying because of the co-existence of other agents (i.e., $A_k(e_{pq}^j) \geq \epsilon_k$ should hold), will contribute to the dissatisfying experience of that agent. Furthermore, each dissatisfying condition is weighted by the edge cost $g(e_{pq}^j)$. In this way, we can quantify the resource experience in terms of how long such a dissatisfying condition lasts in travel time or distance, which is quantified by $g(e_{pq}^j)$. As discussed below, the resource experience of an agent will determine its collision score, which is defined from a probabilistic point of view.

4.1.2. Definition 2 (Collision Score)

We use the resource experience results from Definition 1 to calculate the *collision scores*. This is defined from the view point of collision probability, that must be constrained under some threshold.
Let

- Col_j be the event that agent j announces a collision (i.e., when agent j calls for one of the resources, the allocated resource is less than satisfying);

- $D^j = \{D_1^j, \ldots, D_L^j\}$, where $D_k^j = D\left(\pi^j, A_k\right)$, be the set of dissatisfying experiences of agent j along path π^j on the resource A_k; and

- $f_k \in f = \{f_1, \ldots, f_L\}$ be a customized cumulative distribution function (CDF) defined on $[0, +\infty)$, mapping the resource experience D to a probability of collision on the resource A_k.

The *collision score* of the agent j is defined as the probability of how likely a collision occurs to the agent j on *at least* one of the resources given its resource experience D^j:

$$P\left(Col_j \middle| D^j\right) = 1 - \prod_{k \in \{1,\ldots,L\}} \left(1 - f_k(D_k^j)\right). \tag{8}$$

Note that $P\left(Col_j \middle| D^j\right)$ calculates the *complement* of the success probability—the joint probability of being tolerable at all resources.

Figure 2 shows two example designs of f: $f_1(D) = sigmoid(D - \delta)$, with a discontinuity point $f_1(0) = 0$, is a sigmoid-based CDF function, featuring a surge in the collision score (the derivative is bump-shaped) at the experience value around δ. This function is suitable to important resources that are sensitive to the agent; $f_2(D) = min(1, D/(4\delta))$ is a linear CDF with a shallow slope (the derivative is flat). This function can apply to trivial resources that are not very sensitive to the agent but still accumulate to contribute to the collision score. We use the offset parameter δ to adjust the *tolerance level*

of the dissatisfying experience. With larger δ, the agent will tolerate a longer unpleasant experience before announcing a collision.

Although the definition of the collision score can be customized according to different practices, the probabilistic definition of collision score introduced here is a general one: Different types of resources may have different value ranges, and Equation (8) standardizes the resource ranges, mapping them to a value within $[0,1]$ and enabling an efficient integration of different types of resources to the framework.
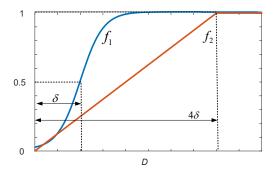
Figure 2. Example designs of cumulative distribution functions (CDFs), mapping the resource experience D of an agent to a collision probability on certain resource. f_1: sigmoid-based CDF for important (sensitive) resources. f_2: linear CDF for trivial (insensitive) resource. δ: offset parameter adjusting the *tolerance level*.

4.1.3. Definition 3 (Soft-Collision Function)

Now, according to the collision scores from Definition 2, we want to pick out the above-threshold agents and place them into the soft-collision set via the *soft-collision function* for the purpose of applying the sub-dimensional expansion.

Given a path $\pi = \pi(v_s, v_b)$ and corresponding resource experience D^j for the agent j, the *soft-collision function* of the agent j is

$$\widetilde{\psi}^j(v_b) = \left\{ \begin{array}{ll} \{j\}, & for \; P\left(Col_j \middle| D^j\right) \geq T \\ \varnothing, & otherwise \end{array} \right. , \tag{9}$$

where T is the *threshold of collision*. The definition of the *global soft-collision function* is then defined as

$$\widetilde{\psi}(v_b) = \bigcup_{j \in I} \widetilde{\psi}^j(v_b). \tag{10}$$

Based on Definition 3, we can formally construct the soft-collision constraint on common resources and obtain the soft-collision constrained MAPP problem:

$$\min_{\pi} g\left(\pi(v_s, v_d)\right)$$
$$s.t. \tag{11}$$
$$\widetilde{\psi}(v_p) = \varnothing, \quad \forall v_p \in \pi.$$

This problem setting is general and can be utilized to express the hard collision setting in Equation (2) by setting $T = 0$ or changing the condition inside the indicator function of Equation (6) to $A_k(e^j_{pq}) \neq A^j_k(e^j_{pq})$ with infinite cost.

4.2. SC-M* Description

SC-M* is a general solver to the MAPP problem in Equation (11) by adjusting M* to the soft-collision constraints on common resources. The pseudocode for SC-M* is presented in Algorithm 2,

where critical commands relative to the soft-collision constraint are underscored. In this algorithm, Lines 1–7 initialize each vertex v in the vertex set V with infinite cost from the source v_s (the cost of v_s itself is zero), set dissatisfying experience to zero and make collision set C_k empty. The initial open list contains v_s only (Line 8). In each iteration, SC-M* expands the first-ranked vertex v_k in the open list ordered by the total cost $v_k.cost + heuristic[v_k]$ (Lines 10 and 11). The algorithm terminates and returns the result if the expanded v_k is the destination v_d (Lines 12–14) or jumps to the next iteration if immediate collision occurs at v_k, i.e., $\widetilde{\psi}(v_k) \neq \varnothing$ (Lines 15–17). Line 18 constructs the *limit neighbors* V_k^{nbh} of v_k using Equation (4). For each vertex v_l in V_k^{nbh} (Line 19), it adds v_l to the descendant set V_k of v_k (Line 20), updates the dissatisfying experience of v_l using Algorithm 1 (Line 21), and merges the immediate collision at v_l to its soft-collision set C_l (Line 22). On top of the new collision set of v_l, SC-M* backpropagates to update all the affected ancestor vertexes from v_l (see Equation (3)) and adds them back to the open list for re-expanding (Line 23). After this collision set updating operation, if v_l is free from collision and has improved cost, the algorithm accepts the new cost by save the trace-back information and adding v_l to the open list for expansion (Lines 24–28). This process repeats until the open list is empty (Line 9) when no solution exists or the optimal solution is found (Lines 12–14).

Algorithm 2 Soft-collision M*.

Input: v_s: source joint vertex; v_d: destination joint vertex; $\{V, E\}$: joint configuration graph;
A: list of resources
Output: Path finding results
1: **for** all v_k in V **do**
2:     $v_k.cost \leftarrow +\infty$
3:     $v_k.exp \leftarrow$ all zero experience
4:     $C_k \leftarrow \varnothing$
5:     $v_k.traceBack \leftarrow \varnothing$
6: **end for**
7: $v_s.cost \leftarrow 0$
8: open $\leftarrow \{v_s\}$
9: **while** open$\neq \varnothing$ **do**
10:     open.sort() by $v.cost+heuristic[v]$ //i.e., sort the open list from small to large
11:     $v_k \leftarrow$ open.pop()
12:     **if** $v_k = v_d$ **then**
13:         **return** back_track_path$[v_k]$ //optimal path found
14:     **end if**
15:     **if** $\widetilde{\psi}(v_k) \neq \varnothing$ **then**
16:         **continue**   //skip the vertex in collisions
17:     **end if**
18:     conduct the construction of V_k^{nbh} using Equation (4)
19:     **for** v_l in V_k^{nbh} **do**
20:         add v_l to V_k //note $V_k = \{v_q | \exists \pi(v_k, v_q) \subseteq G^{exp}\}$
21:         $v_l \leftarrow$ experience(v_k, v_l, A) //update experience using Algorithm 1
22:         $C_l \leftarrow C_l \cup \widetilde{\psi}(v_l)$
23:         backpro_update$(v_k, C_l,$ open)   // 1) update all the affected soft-collision sets using Eq.(3)
                                        //2) add all affected vertexes back to open list (see reference [6] for details)
24:         **if** $\widetilde{\psi}(v_l) = \varnothing$ **and** $v_k.cost+e_{kl}.cost < v_l.cost$ **then**
25:             $v_l.cost \leftarrow v_k.cost+e_{kl}.cost$
26:             $v_l.traceBack \leftarrow v_k$
27:             open.add(v_l)
28:         **end if**
29:     **end for**
30: **end while**
31: **return** no path exists

SC-M* can make a transition from a decoupled individual A* ($T = 1$) to a standard hard-collision constrained M* ($T = 0$), providing more flexibility to the performance of the algorithm with bounded soft-collision scores.

4.3. Completeness and Cost-Suboptimality

A MAPP algorithm is complete if it guarantees that it will either return a path, or determine that no path exists in finite time. An algorithm is optimal if it guarantees returning an optimal path if such a solution exists. SC-M* is complete and suboptimal conditioned on the soft-collision constraint (i.e., $P\left(Col_j|D^j\right) < T$, for a given collision threshold T).

4.3.1. Completeness

Theorem 1. *SC-M* is a complete algorithm.*

Proof of Theorem 1. SC-M* inherits the sub-dimensional expansion from M* (i.e., it changes the G^{sch} only when one of the soft-collision sets C_p changes). The algorithm applies A* in the updated search graph. Due to the merging operation applied to collision set C_p, as shown in Equation (3), C_p for each vertex will change finite times (at most m times, which is the number of agents). Because A* is complete, applying A* to a given G^{sch} takes finite time to return a result. Therefore, SC-M* is complete. □

4.3.2. Cost-Suboptimality

Different from M*, which is optimal, SC-M* is suboptimal because Equations (9) and (10) only include the immediate conflicting agents to the soft-collision set; the agents that softly interfere with the conflicting agents in the upstream path are excluded. Those excluded agents also contribute to the announced collision (i.e., making the collision score above the threshold). Because of this, SC-M* cannot guarantee the *inclusion property*, which is the basis to ensure the optimality in M* [6]: *The optimal path for some subset of agents costs no more than the optimal joint path for the entire set of agents.* Without the inclusion property, SC-M* may not guarantee cost optimality.

Figure 3 provides a counterexample of the inclusion property of SC-M* in the soft-collision MAPP context defined in this paper. Let $\pi'_\Omega(v_k, v_f)$ be the joint path constructed by combining the optimal path for a subset $\Omega \in I$ of agents with the individually optimal paths for the agents in $I\backslash\Omega$. The inclusion property is defined as follows: If the configuration graph contains an optimal path $\pi_*(v_k, v_f)$, then $\forall\Omega \subset I$, $g(\pi'_\Omega(v_k, v_f)) \leq g(\pi_*(v_k, v_f))$. See *Lemma 6* in [6].

In the soft-collision context, this inclusion property does not always hold. In Figure 3, we have a three-agent MAPP problem ($I = \{r1, r2, r3\}$) in the soft-collision context. Agents $r1$, $r2$, and $r3$ attempt to move from the vertexes a, f, and h to the vertexes e, g, and i, respectively. The individually optimal paths (shortest distance) are $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ with distance 4 for $r1$, $f \rightarrow c \rightarrow d \rightarrow g$ with distance 3 for $r2$, and $h \rightarrow b \rightarrow c \rightarrow i$ with distance 3 for $r3$. The total cost of the joint individually optimal path is 10. However, assuming that the agents can only tolerate a dissatisfying experience with distance 1, $r1$ will announce a collision at vertex d because of the interference on the edge $b \rightarrow c$ and $c \rightarrow d$ from agents $r3$ and $r2$, respectively.

If we choose $\Omega = \{r1, r2\} \in I$, as can be seen in Figure 3, the only solution would be that $r1$ takes a detour through the vertex x to avoid the collision on the edge $c \rightarrow d$, resulting in a cost of 5 for $r1$, and the total $g(\pi'_\Omega(v_k, v_f))$ is 11 (3 for $r2$, 5 for $r1$ and 3 for $r3$). On the other hand, by searching through all three dimensions, a better solution would be that $r3$ detours through the vertex y, and $r1$ is free from collision because the interference on the edge $b \rightarrow c$ disappears. The total cost of this joint path is 10.5, and we have $g(\pi'_\Omega(v_k, v_f)) = 11 > g(\pi_*(v_k, v_f)) = 10.5$, which is contradictory to the inclusion property.
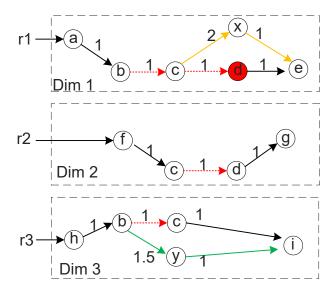
Figure 3. Counterexample of the inclusion property of soft-collision M* (SC-M*) in the soft-collision context. Agents $r1, r2$, and $r3$ have the planning O-D demands $(a, e), (f, g)$, and (h, i), respectively. Vertexes in the system are labeled as a, b, c, etc.

The reason for this phenomenon is that, in the hard-collision context, only the immediate conflicting agent $r2$ contributes to the collision of $r1$ at vertex d. However, in the soft-collision context, both $r2$ and $r3$ contribute to the collision of $r1$ at vertex d, and thus, the inclusion property does not apply. Without this inclusion property, which is the basis of the optimality of M*, the optimality of SC-M* cannot be guaranteed.

However, we notice that suboptimal methods have long been used successfully to solve many interesting MAPP problems [15,25,26]. Given the fact that we show in the next section that SC-M* is superior to other alternative SC-based MAPP solvers (e.g., SC-A* and SC-CBS) in terms of scalability, run time, and path cost, we demonstrate that the proposed method, which is adjusted to MAPP in the soft-collision context, is a powerful tool in practice.

5. Experiments and Results

We evaluated SC-M* in simulation on a grid public mass transit network with an Intel Core i7-6700 CPU at 3.4 GHz with 16 GB RAM. As shown in Figure 4, the grid transit environment has 20×20 stops. There are 20 bidirectional horizontal lines. Likewise, 20 bidirectional vertical lines are deployed in the environment. At each stop, agents can switch lines. The yellow areas are covered by some resources, such as the on-vehicle free Wi-Fi in our experiments. Agents traversing those areas can enjoy high-quality on-vehicle Wi-Fi connections. A fully covered edge has a Wi-Fi resource value of 100, and the Wi-Fi value of an edge is proportional to the length of coverage. Each agent wants to move from its source (square) to its destination (circle) with the lowest cost (i.e., a linear combination of distance cost and Wi-Fi cost) as well as bounded collision score. The second resource is the space on the edge, which is fixed at 5. The satisfying values are $\varepsilon_1 = 20$ and $\varepsilon_2 = 1$ for Wi-Fi and space resources, respectively.

We randomly generated a source–destination pair for each agent. Each trial was given a 1000-s run-time limitation to find a solution. For each configuration (including the number of agents, collision threshold T, and offset parameter δ), we ran 20 random trials to calculate the average metrics (i.e., the success rate and run time). The success rate is the number of trials ending with a solution divided by the number of trials. The run time is the average over trials ending with a solution or a no-solution declaration. If all trials under a certain configuration exceeded 1000 s, we used ">1000" to represent the run time of the corresponding configuration. We used the standard A* as the coupled planner and policy generator in the SC-M* framework and compared our results to the baselines.
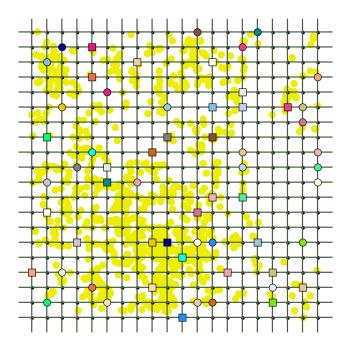
Figure 4. Grid system with 20×20 stops and 40 bidirectional lines. Square and circle of the same color correspond to the source and destination of an agent, respectively.

5.1. Planning for the One-Resource-One-Agent-Type

The first experiment considered Wi-Fi as the only resource requested by agents (i.e., $A = \{A_1 : "WiFi"\}$). Only one agent type exists, and all agents use sigmoid-based function f_1 as the collision CDF.

We first studied the influence of the collision threshold T and the offset parameter δ on performance. Figure 5a shows the success rate of the one-resource-one-type SC-M* with different thresholds $T = 0$ (equivalent to the basic M*), 0.2, 0.4, and 0.45, while the offset parameter is fixed to $\delta = 6.0$. Table 1 (left) shows the run time in seconds for the experiment. The results clearly show that larger thresholds bring improvement in performance with a higher success rate and lower run time for a large system size ($m > 50$). The improvement in performance results from the property of SC-M* that larger thresholds render more relaxed constraints, and thus, agents are less likely to collide on resources.
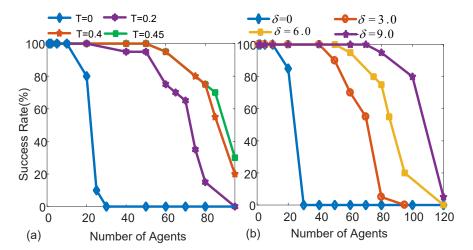

Figure 5. Impact of the collision threshold T (given $\delta = 6.0$) and offset parameter δ (given $T = 0.35$) on one-resource-one-type SC-M*.

Figure 5b shows the success rate of the SC-M* with different offset parameters $\delta = 0, 3.0, 6.0,$ and 9.0, with fixed $T = 0.35$. Table 1 (right) shows the run time for the experiment. The results

illustrate that SC-M* is sensitive to δ and can efficiently handle up to 100 agents with $\delta = 9.0$. These results are reasonable because the sigmoid-based CDF is used in the experiments, featuring a surge in the collision probability at the experience value around the offset, and the offset parameter poses a cutoff value on the resource experience, with collision always announced once the resource experience is larger than the offset. The standard M* ($T = 0$) can only scale to fewer than 30 agents. Taking advantage of this property, one can tune the parameters to trade off the scalability against the tightness of constraints.

Table 1. Run time of one-resource-one-type SC-M* under different parameters.

m	$T = 0$	$T = 0.2$	$T = 0.4$	$T = 0.45$	m	$\delta = 3.0$	$\delta = 6.0$	$\delta = 9.0$
5	0.556389	0.52489	0.5472	0.3616	5	0.506	0.3616	0.575
10	1.25143	1.18687	0.7057	0.7965	10	1.0765	0.7965	1.0427
20	403.3011	2.72513	1.4871	1.4488	20	2.2578	1.4488	2.1034
40	>1000	56.1898	4.2336	4.4318	40	17.201	4.4318	4.525
70	>1000	370.059	257.59	255.78	80	477.96	292.17	59.31
95	>1000	>1000	951.34	774.40	120	>1000	>1000	857.0
	Left: $\delta = 6.0$					Right: $T = 0.35$		

5.2. Planning for the Two-Resource-Two-Agent-Type

We also evaluated SC-M* in more complex environments: two agent types requesting two resources. This experiment considered both Wi-Fi and space capacity (i.e., $A = \{A_1 : \text{"WiFi"}, A_2 : \text{"Space"}\}$). Type I agents use f_1 in Figure 2 as the collision CDF for the Wi-Fi resource, and the linear CDF f_2 for the space resource, implying that they treat Wi-Fi and space as important and trivial, respectively. On the other hand, Type II agents use f_1 for space and f_2 for Wi-Fi. Each agent has a 50% chance of being Type I. Both CDFs are adjusted using the same δ at each trial, as illustrated in Figure 2.

Figure 6a shows the success rate of the two-resource-two-type SC-M* with different thresholds $T = 0$ (equivalent to the basic M*), 0.2, 0.35, and 0.45, and with a fixed offset parameter $\delta = 9.0$. Table 2 (left) shows the run time for the experiments. As can be seen from the results, in general, SC-M* can handle the two-resource-two-type systems and plan for more than 80 agents. Because more resources contribute more factors to increasing the collision score, a relatively large offset ($\delta = 9.0$) is needed to achieve comparable performance to the one-resource-one-type SC-M*.
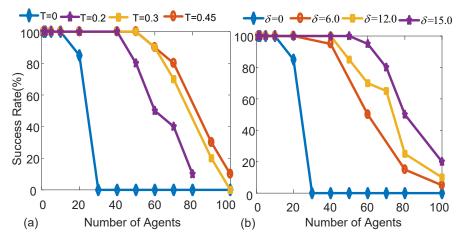

Figure 6. Impact of the collision threshold T (given $\delta = 9.0$) and offset parameter δ (given $T = 0.35$) on two-resource-two-type SC-M*.

Table 2. Run time of two-resource-two-type SC-M* under different parameters.

m	*T* = 0.2	*T* = 0.35	*T* = 0.45	*m*	*δ* = 6.0	*δ* = 12.0	*δ* = 15.0
5	0.3438	0.3493	0.363948	5	0.333498	0.3677	0.527464
20	1.2485	1.8549	1.807993	20	1.479236	1.4032	2.369483
40	10.102	3.2387	4.415021	40	61.49792	4.5024	4.04331
60	503.94	106.02	104.6499	60	521.2721	306.46	60.30944
90	901.91	801.47	702.4526	70	627.0925	347.14	209.5725
100	>1000	909.0	901.1799	100	901.91	751.78	606.6522
Left: *δ* = 9.0				Right: *T* = 0.35			

Figure 6b and Table 2 (right) present the impact of the offset parameter *δ* on performance. Different from the first experiment, SC-M* with the above configurations is less sensitive to *δ*, when compared to Figure 5. The reason is that 50% of the agents are insensitive to one of the resources because of the linear CDF f_2, thus increasing *δ* does not contribute to a significant reduction in collisions. This property implies that we can control the importance levels of resources efficiently through the design of collision CDFs. This experiment demonstrates that, with the proper parameter settings, SC-M* can feasibly handle a complex environment with multiple resources and multiple agent types.

5.3. Comparison of SC-M to Baselines*

We next compared the SC-M* to other SC-based MAPP algorithms, including SC-A* (optimal) and SC-CBS (suboptimal), in the one-resource-one-type environment.

5.3.1. Path Cost

Firstly, we compared the path cost of the three algorithms. We designed 60 planning tasks for environments with 4–6 agents (20 tasks for each), in which agents will encounter at least one collision along the individually optimal paths under the $T = 0.05$, $δ = 1$ setting. We start with small agent numbers because SC-A* cannot handle a large number of agents.

Figure 7 shows the average difference of the three SC-based solvers relative to the individually optimal cost (i.e., the sum of the optimal cost of each agent when the agent is the only one in the system). In other words, the Y-axis represents the cost of collisions. We observe that SC-A* and SC-CBS have the lowest and highest additional cost, respectively. SC-M* solutions cost more than SC-A* but noticeably less than SC-CBS.
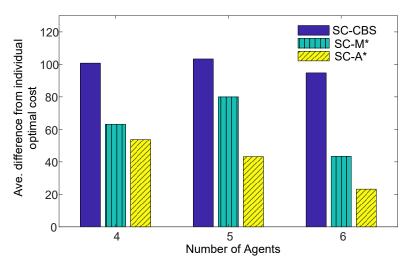

Figure 7. Average cost difference of soft-collision-based multi-agent path planning (SC-based MAPP) solvers from the individually optimal cost in the one-resource-one-type context.

To be more detailed, in the experiments, we designed MAPP tasks for environments containing 4–6 agents with 20 tasks for each. All tasks were designed to encounter at least one collision along the individually optimal paths under the above-mentioned configuration. Thus, additional costs relative to the individually optimal path are expected for each of the three SC-based MAPP solvers. Table 3 compares the results of SC-M* and SC-CBS to the optimal solutions obtained by SC-A*. The top half of the table shows the increase in cost relative to the cost for SC-A*; the costs for SC-A* for all scenarios vary within a small range so the results are in absolute numbers. The bottom half shows the ratio in run time with respect to SC-A*; the run time for SC-A* varies greatly across the experiments so we show the cost reduction as a percentage. In the table, we observe that the additional cost of SC-M* from the SC-A* is consistently lower than that of SC-CBS. We also observe that SC-M* is significantly faster than SC-A* and competitive relative to the run time of SC-CBS. The standard deviations show the fluctuations of the solutions for SC-M* and SC-CBS around the optimal solutions for SC-A*.

Table 3. Results of the path cost experiments.

| | idx | *m* = 4 | | *m* = 5 | | *m* = 6 | |
		SC-CBS	SC-M*	SC-CBS	SC-M*	SC-CBS	SC-M*
	1	50.60	0.00	287.00	216.60	22.00	0.00
	2	11.00	0.00	5.50	0.00	50.60	50.60
	3	1.10	1.10	45.10	7.70	5.50	0.00
	4	136.50	0.00	6.60	0.00	11.00	0.00
	5	62.70	31.80	81.40	81.40	177.10	0.00
	6	9.90	9.90	38.93	34.31	0.00	0.00
	7	22.00	22.00	270.50	204.50	33.14	26.65
Cost	8	16.50	16.50	3.30	3.30	182.18	0.00
difference	9	13.20	0.00	27.50	0.00	211.10	169.30
from	10	104.50	0.00	78.30	67.08	32.33	20.89
SC-A*	11	58.08	24.08	22.00	0.00	79.20	0.00
	12	20.90	0.00	36.84	5.94	52.80	15.40
	13	11.00	0.00	115.40	56.00	72.16	59.73
	14	28.60	28.60	17.15	9.34	63.70	26.84
	15	13.20	0.00	66.00	0.00	19.11	14.05
	16	94.60	0.00	35.53	23.65	18.70	1.10
	17	16.83	2.86	14.90	10.71	318.90	0.00
	18	205.70	22.00	1.10	1.10	48.86	8.94
	19	53.24	27.71	14.90	10.71	1.10	1.10
	20	12.10	3.30	34.75	2.53	32.61	8.62
Std. dev		52.44	12.00	80.49	64.14	84.64	39.15
	1	38.39%	52.84%	24.54%	18.24%	0.33%	0.58%
	2	10.41%	20.12%	4.37%	3.42%	0.23%	0.26%
	3	14.53%	24.58%	0.09%	0.30%	2.32%	4.60%
	4	25.12%	15.42%	0.29%	0.71%	0.92%	1.35%
	5	0.81%	0.54%	21.62%	56.30%	0.55%	0.82%
	6	19.17%	15.67%	0.52%	0.59%	3.48%	3.07%
	7	14.02%	23.87%	16.39%	17.05%	0.43%	0.44%
Run time	8	29.82%	19.11%	1.40%	1.89%	0.07%	0.18%
percentage	9	64.67%	14.87%	6.16%	8.93%	0.13%	0.14%
to	10	46.67%	37.19%	0.45%	0.73%	0.10%	0.15%
SC-A*	11	5.94%	16.49%	1.96%	5.56%	0.15%	0.23%
	12	5.16%	36.22%	2.36%	3.29%	0.46%	1.19%
	13	11.40%	19.94%	0.44%	1.12%	0.47%	0.61%
	14	6.18%	17.72%	0.76%	2.11%	0.78%	0.58%
	15	33.07%	44.81%	25.38%	34.96%	0.48%	0.56%
	16	86.83%	37.86%	1.92%	3.10%	0.26%	0.20%
	17	16.10%	29.91%	0.92%	2.10%	4.01%	3.95%
	18	11.68%	31.59%	10.04%	9.57%	0.28%	0.67%
	19	3.97%	14.12%	0.92%	2.10%	1.67%	1.80%
	20	4.37%	13.83%	1.01%	1.54%	0.60%	0.52%
Std. dev.		22.34%	12.57%	8.65%	14.08%	1.12%	1.30%

The reason for the results is that SC-A* is an optimal solver for this type of MAPP problem because it always explores cheaper paths in the entire multi-agent joint space before considering the paths that cost more [7]. SC-M* is suboptimal because of the process discussed in Section 4.3.2. Compared to SC-M*, SC-CBS suffers from more path cost due to the way it collects a collision: CBS collects collisions into a *conflict tree* and arranges the collision into the form [agent j, vertex v, step s], indicating that agent j collides at vertex v at step s. In each iteration, CBS conducts decoupled planning to avoid agent j reaching vertex v at step s. This might lose some information in the soft-collision context because there might exist another path that leads j to vertex v at step s without announcing a collision, by avoiding one of the upstream vertexes involved in soft interference. In contrast, SC-M* can explore those paths excluded by SC-CBS because it searches the entire space of the immediate colliding agents. Figure 8 provides an example to visualize the difference in planning among the three SC-based MAPP solvers.
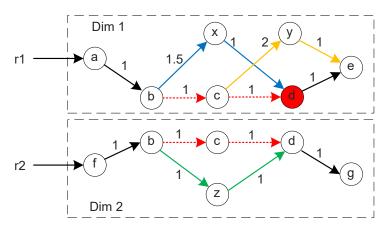

Figure 8. Illustration of the difference in planning among soft-collision A* (SC-A*), soft-collision M* (SC-M*), and soft-collision conflict-based search (SC-CBS).

Figure 8 shows a two-agent MAPP problem in the soft-collision context. Agents $r1$ and $r2$ attempt to move from vertexes a and f to vertexes e and g, respectively. The individually optimal paths (shortest distance) for both agents are $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ with distance 4 for $r1$ and $f \rightarrow b \rightarrow c \rightarrow d \rightarrow g$ with distance 4 for $r2$, respectively. The total cost of the joint individually optimal path is 8. $r1$ and $r2$ softly collide on the edge $b \rightarrow c$ and $c \rightarrow d$, where $r2$ can tolerate the dissatisfying experience with distance 2. However, $r1$ can only tolerate the dissatisfying experience with distance 1 and announces a collision at the vertex d.

When using SC-CBS, we record the collision that occurred to $r1$ as [$r1$, d, 3], indicating that agent $r1$ will collide at vertex d at the third step. Then, SC-CSB will avoid any paths leading $r1$ to d at Step 3 (including $a \rightarrow b \rightarrow x \rightarrow d \rightarrow e$ and $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$) and will end up with a longer detour through vertex y. The SC-CBS solution has a cost of 5 for $r1$ and 9 in total.

When using SC-M*, the collision at d triggers the sub-dimensional expansion of the search graph in dimension 1, which includes both x and y. Thus, it can find a cheaper collision-free path through x and end up with a path $a \rightarrow b \rightarrow x \rightarrow d \rightarrow e$ with a dissatisfying experience of distance 1 and a cost of 4.5 for $r1$ (8.5 in total). However, SC-M* does not expand dimension 2 because no collision has been announced by $r2$.

When using SC-A*, the joint search space of both dimension 1 and dimension 2 is expanded and searched. Instead of vertexes x and y, SC-A* will first investigate vertex z in dimension 2 according to some heuristics. This process leads to another cheaper path $f \rightarrow b \rightarrow z \rightarrow d \rightarrow g$ with distance 4 for $r2$ (8 in total, which is the same as the individually optimal cost) and avoids all interference by moving through this path. As a result, SC-A* returns an optimal solution that satisfies the soft-collision constraint at the expense of search space.

The example in Figure 8 illustrates the optimality of SC-A* and the advantage of SC-M* in path cost over SC-CBS. To be specific, SC-M* provides a better solution than SC-CBS by searching thoroughly

through the expanded dimensions, whereas the way SC-CBS identifies collisions is inappropriate in the soft-collision context. To the best of our knowledge, no other methodology capable of dealing with the soft-collision path planning defined in Equation (11) has been developed. It is expected that, in the future, more high-performance algorithms will be developed for solving the problem.

5.3.2. Run Time

Table 4 shows the average run time of the three SC-based MAPP solvers and we observe that both SC-M* and SC-CBS are significantly faster than SC-A* in terms of run time. This is reasonable because SC-A* always searches the global high-dimensional joint space, which is expensive. SC-CBS is faster than SC-M* because it always searches in one individual dimension at a time, whereas the SC-M* needs to occasionally deal with high-dimensional space when collisions occur.

Table 4. Average run time of SC-based MAPP solvers in the one-resource-one-type context.

m	SC-CBS	SC-M*	SC-A*
4	1.971	2.002	47.35
5	1.798	3.312	473.7
6	1.942	2.969	390.0

5.3.3. Scalability

We compared the scalability of the three SC-based MAPP solvers in terms of planning for a large system size ($m > 50$). Figure 9 presents the success rate, average additional cost (i.e., how much more cost than the individually optimal path), and run-time ratio over SC-CBS under different thresholds T, where the run-time ratio of SC-CBS is compared to itself and thus is constant. SC-A* has the slackest constraint ($T = 0.35, \delta = 9.0$) but poorest performance because of the prohibitively large search space. SC-CBS has the best success rate because of the property of the decoupled searching. However, this is at the expense of path cost. SC-M* performs decently in terms of both the success rate (significantly superior to SC-A*) and cost (noticeably lower than SC-CBS) as the number of agents increases.
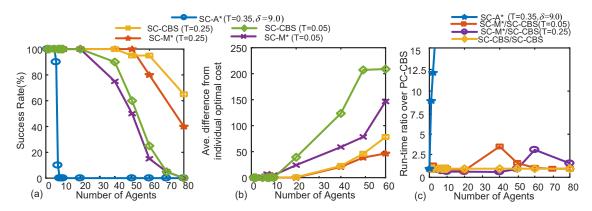

Figure 9. Success rate, cost, and run time ratio of the three SC-based MAPP solvers under different T.

The run time of the SC-M* is generally longer than that of SC-CBS. In another experiment, we observe that the run-time ratio of SC-M* over that of the SC-CBS starts to decrease after a peak. This is because we force all algorithms to terminate after 1000 s, and both curves will converge to value one when their success rates decline to zero. We conducted another scalability experiment with different offsets δ (given $T = 0.25$) and observe the same results in terms of scalability. Figure 10 shows the experimental results.
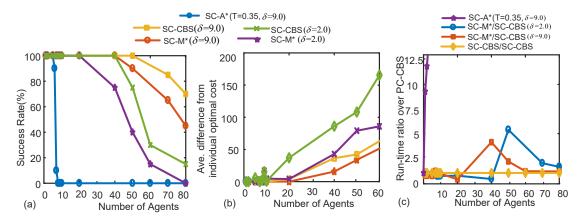
Figure 10. Success rate, cost, and run-time ratio of the three SC-based MAPP solvers under different δ.

Considering the scalability and path cost altogether, SC-M* demonstrates its overall advantages over alternative SC-based solvers.

6. Conclusions

This paper proposes SC-M*, a generalized version of M* with soft-collision constraints on common resources, which can scale to solving the multi-agent path planning problem in the soft-collision context. The SC-M* tracks the collision score of each agent and place agents, whose collision scores exceed some thresholds into a soft-collision set for sub-dimensional expansion. We show that the SC-M* has advanced flexibility and scalability for efficiently solving MAPP problems in the soft-collision context and can handle complex environments (e.g., with multiple types of agents requesting multiple types of resources). We compare the SC-M* to other SC-based MAPP solvers and show the advantages and trade-offs of the SC-M* against baselines in terms of path cost, success rate, and run time.

Future work will focus on leveraging advanced variants of M*, such as EPErM*, ODrM*, etc., to remove the basic A* component in our planner. We believe that better performance can be obtained this way because these variants improve the coupled planner and policy generator (two important components in the basic M*), which are directly related to the M* bottlenecks that limit the planning scalability. We are also interested in applying SC-M* to real-world applications for case studies. One promising research direction is to use the proposed algorithm to serve the passengers in public transits. It is expected that SC-M* will handle large-scale mobility demands in cities

Author Contributions: Conceptualization, R.S., P.S. and M.M.V.; Methodology, R.S. and M.M.V.; experimental design, R.S. and P.S.; software, R.S.; analysis, R.S.; writing, original draft preparation, R.S.; writing, review and editing, R.S., P.S. and M.M.V.; supervision, M.M.V. and P.S.; and funding acquisition, M.M.V. and P.S.

Abbreviations

The following abbreviations are used in this manuscript:

MAPP	Multi-Agent Path Planning
SC-M*	Soft-Collision M*
OD	Operator Decomposition
EPEA*	Enhanced Partial Expansion A*
IDA*	Iterative Deepening A*
CBS	Conflict-Based Search
MA-CBS	Meta-Agent Conflict-Based Search

References

1. Dresner, K.; Stone, P. A multiagent approach to autonomous intersection management. *J. Artif. Intell. Res.* **2008**, *31*, 591–656. [CrossRef]
2. Pallottino, L.; Scordio, V.G.; Bicchi, A.; Frazzoli, E. Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Trans. Robot.* **2007**, *23*, 1170–1183. [CrossRef]
3. Silver, D. Cooperative Pathfinding. In Proceedings of the 1st Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), Marina del Rey, CA, USA, 1–3 June 2005; pp. 117–122.
4. Wagner, G.; Choset, H. M*: A complete multirobot path planning algorithm with performance bounds. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, USA, 25–30 September 2011; pp. 3260–3267.
5. Ratner, D.; Warmuth, M.K. Finding a shortest solution for the NxN extension of the 15-PUZZLE is intractable. In Proceedings of the 5th AAAI Conference on Artificial Intelligence (AAAI), Philadelphia, PA, USA, 11–15 August 1986; pp. 168–172.
6. Wagner, G.; Choset, H. Subdimensional expansion for multirobot path planning. *Artif. Intell.* **2015**, *219*, 1–24. [CrossRef]
7. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
8. Standley T.S. Finding optimal solutions to cooperative pathfinding problems. In Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI), Atlanta, GA, USA, 11–15 July 2010; pp. 173–178.
9. Felner, A.; Goldenberg, M.; Sharon, G.; Stern, R.; Beja, T.; Sturtevant, N.; Schaeffer, J.; Holte, R. Partial-expansion A* with selective node generation. In Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI), Toronto, ON, Canada, 22–26 July 2012; pp. 471–477.
10. Goldenberg, M.; Felner, A.; Stern, R.; Sharon, G.; Sturtevant, N.; Holte, R.C.; Schaeffer, J. Enhanced partial expansion A*. *J. Artif. Intell. Res.* **2014**, *50*, 141–187. [CrossRef]
11. Korf, R.E. Depth-first iterative-deepening: An optimal admissible tree search. *Artif. Intell.* **1985**, *27*, 97–109. [CrossRef]
12. Sanchez, G.; Latombe, J.C. Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, USA, 11–15 May 2002; pp. 2112–2119.
13. Sharon, G.; Stern, R.; Felner, A.; Sturtevant, N.R. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* **2015**, *219*, 40–66. [CrossRef]
14. Ferner, C.; Wagner, G.; Choset, H. ODrM* optimal multirobot path planning in low dimensional search spaces. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 3854–3859.
15. Wagner, G.; Choset, H. Path planning for multiple agents under uncertainty. In Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS), Pittsburgh, PA, USA, 18–23 July 2017; pp. 577–585.
16. Ma, H.; Kumar, T.S.; Koenig, S. Multi-agent path finding with delay probabilities. In Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI), San Francisco, CA, USA, 4–9 Feburary 2017; pp. 3605–3612.
17. Shi, R.; Steenkiste, P.; Veloso, M.M. Second-order destination inference using semi-supervised self-training for entry-only passenger data. In Proceedings of the 4th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT), Austin, TX, USA, 5–8 December 2017; pp. 255–264.
18. Shi, R.; Steenkiste, P.; Veloso, M.M. Generating synthetic passenger data through joint traffic-passenger modeling and simulation. In Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 3397–3402.
19. Shi, R. Optimizing Passenger on-Vehicle Experience Through Simulation and Multi-Agent Multi-Criteria Mobility Planning. Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, May 2019.
20. Wang, H.; Xie, H.; Qiu, L.; Yang, Y.R.; Zhang, Y.; Greenberg, A. COPE: Traffic engineering in dynamic networks. In Proceedings of the 2006 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM), Pisa, Italy, 11–15 September 2006; pp. 99–110.

21. Fletcher, L.; Teller, S.; Olson, E.; Moore, D.; Kuwata, Y.; How, J.; Leonard, J.; Miller, I.; Campbell, M.; Huttenlocher, D.; et al. The MIT—Cornell Collision and Why It Happened. *J. Field Robot.* **2008**, *25*, 775–807. [CrossRef]

22. Leonard, J.; How, J.; Teller, S.; Berger, M.; Campbell, S.; Fiore, G.; Fletcher, L.; Frazzoli, E.; Huang, A.; Karaman, S.; et al. A perception-driven autonomous urban vehicle. *J. Field Robot.* **2008**, *25*, 727–774. [CrossRef]

23. Zhou, W.; Zhang, C.; Wang, Q. Optimal flow distribution of military supply transportation based on network analysis and entropy measurement. *Eur. J. Oper. Res.* **2018**, *264*, 570–581. [CrossRef]

24. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]

25. Jansen, M.R.; Sturtevant, N.R. Direction Maps for Cooperative Pathfinding. In Proceedings of the 4th Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), Stanford, CA, USA, 22–24 October 2008; pp. 185–190.

26. Van Den Berg, J.; Abbeel, P.; Goldberg, K. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *Int. J. Robot. Res.* **2011**, *30*, 895–913. [CrossRef]