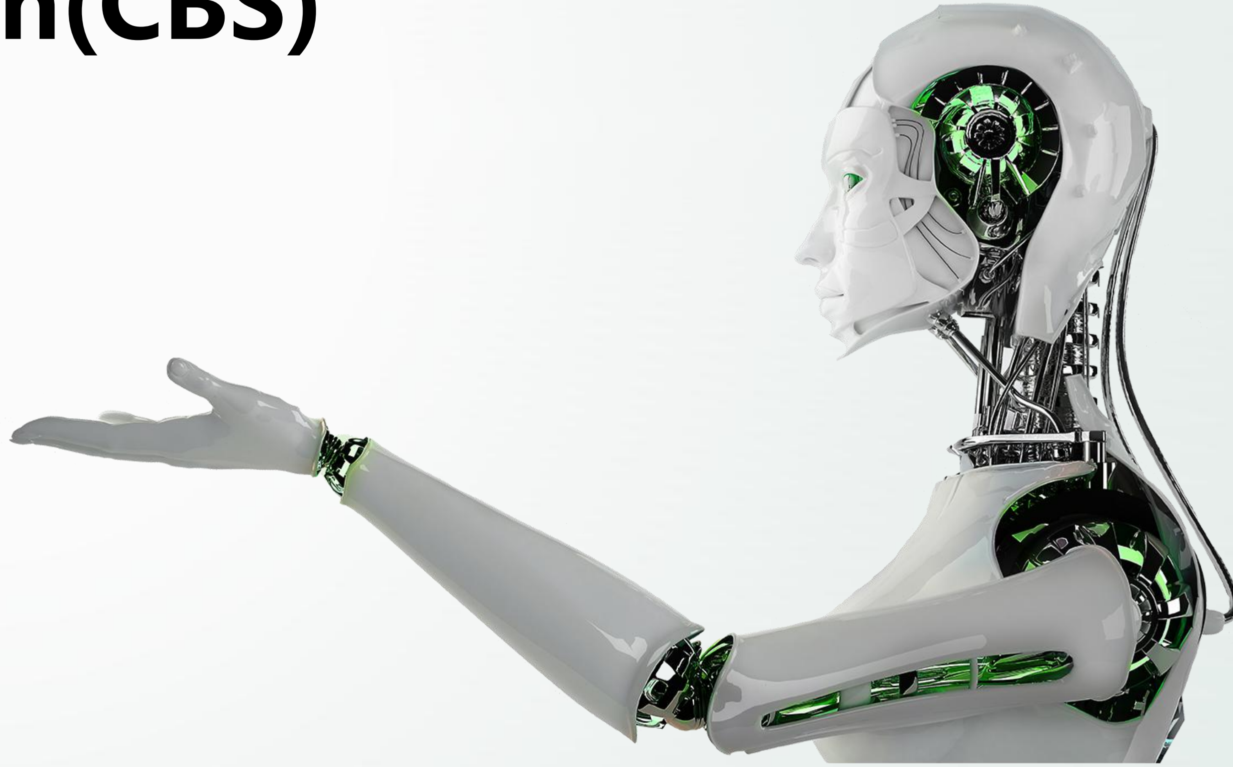
睿慕课 AIIMOOC.COM

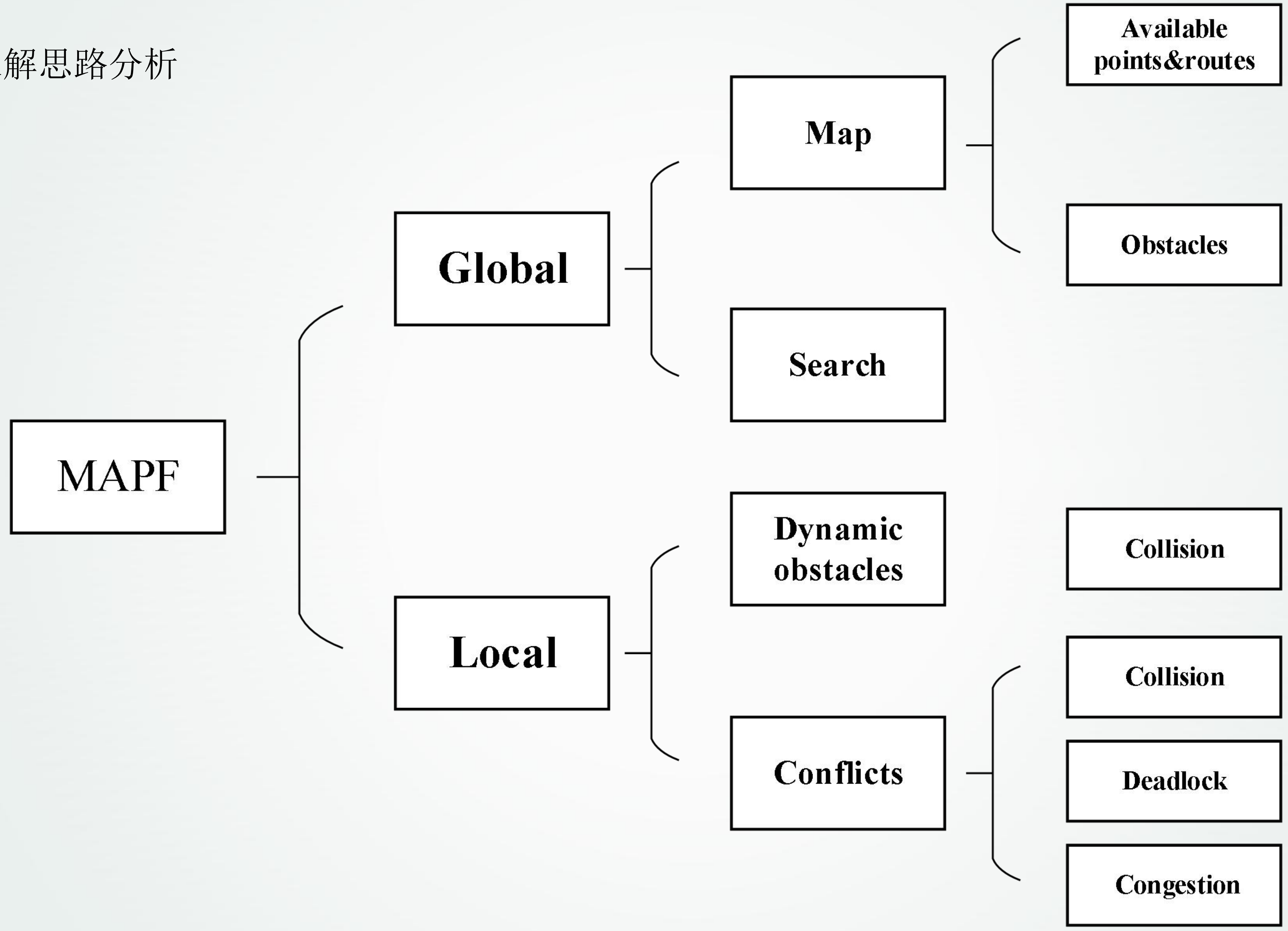
第七章

Conflict-based Search(CBS)

1. CBS原理
2. CBS的优势与缺陷
3. 改进CBS思路



MAPF求解思路分析


MAPF
Global
Local
Map
Search
Dynamic obstacles
Conflicts
Available points&routes
Obstacles
Collision
Collision
Deadlock
Congestion

1. CBS原理

什么是CBS？

CBS和第六章的PP（基于优先级的规划）有何联系？

CBS和三~六章讲到的其他算法有何联系？

1. CBS原理

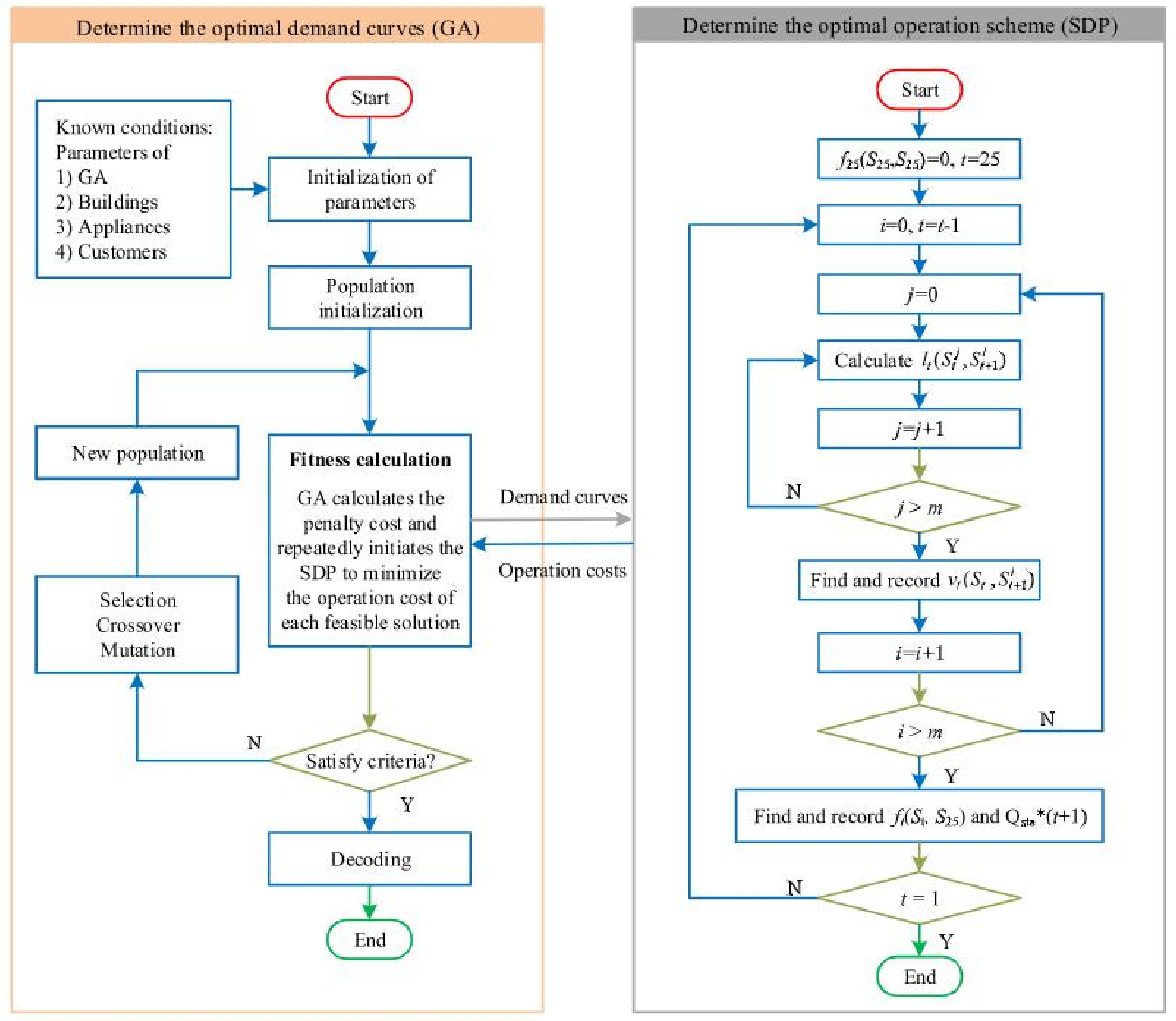
Conflict-based Search (CBS)，基于冲突的搜索，是一种两层算法。CBS包含两层，低层次和高层次。低级搜索用来查找单个代理路径，高级搜索用来查找和解决路径之间的冲突。

双层优化是一种具有二层递阶结构的优化方法，上层和下层都有各自的决策变量，约束及目标。

上层决策者通过自己的决策去指导或命令下层决策者，并不直接干涉下层的决策过程；下层决策者以上层的决策为参数，在自己的可控范围内自由决策。

本质上仍然是单层优化算法

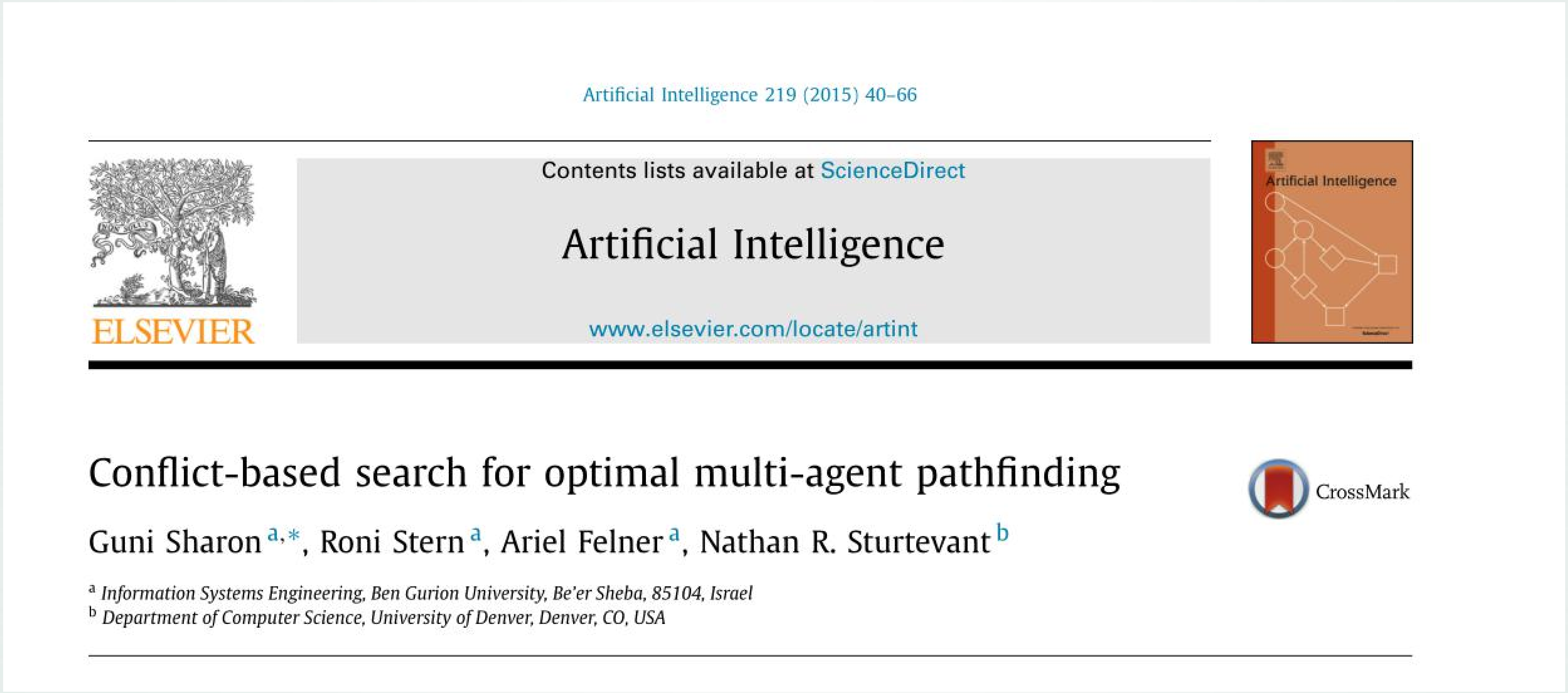
1. CBS原理



Determine the optimal demand curves (GA)
Start
Known conditions: Parameters of 1) GA 2) Buildings 3) Appliances 4) Customers
Initialization of parameters
Population initialization
New population
Fitness calculation
GA calculates the penalty cost and repeatedly initiates the SDP to minimize the operation cost of each feasible solution
Selection Crossover Mutation
N
Satisfy criteria?
Y
Decoding
End
Demand curves
Operation costs
Determine the optimal operation scheme (SDP)
Start
$f_{25}(S_{25}, S_{25})=0$, $t=25$
$i=0$, $t=t-1$
$j=0$
Calculate $l_t(S_t^i, S_{t+1}^j)$
$j=j+1$
N
$j > m$
Y
Find and record $v_t(S_t, S_{t+1}^j)$
$i=i+1$
$i > m$
N
Y
Find and record $f_t(S_t, S_{25})$ and Q_{sts}*$(t+1)$
N
$t = 1$
Y
End

Fig. 4. Flowchart for solving the two-stage optimization model.

1. CBS原理



Artificial Intelligence 219 (2015) 40–66

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

ELSEVIER

Conflict-based search for optimal multi-agent pathfinding

CrossMark

Guni Sharon [a,*], Roni Stern [a], Ariel Felner [a], Nathan R. Sturtevant [b]

[a] *Information Systems Engineering, Ben Gurion University, Be'er Sheba, 85104, Israel*
[b] *Department of Computer Science, University of Denver, Denver, CO, USA*

1. CBS原理

Input ： 有向图G（V,E）；n个机器人组成集群，每个机器人设置起点及终点

Action： a) idle

b) busy: move, wait(stay)

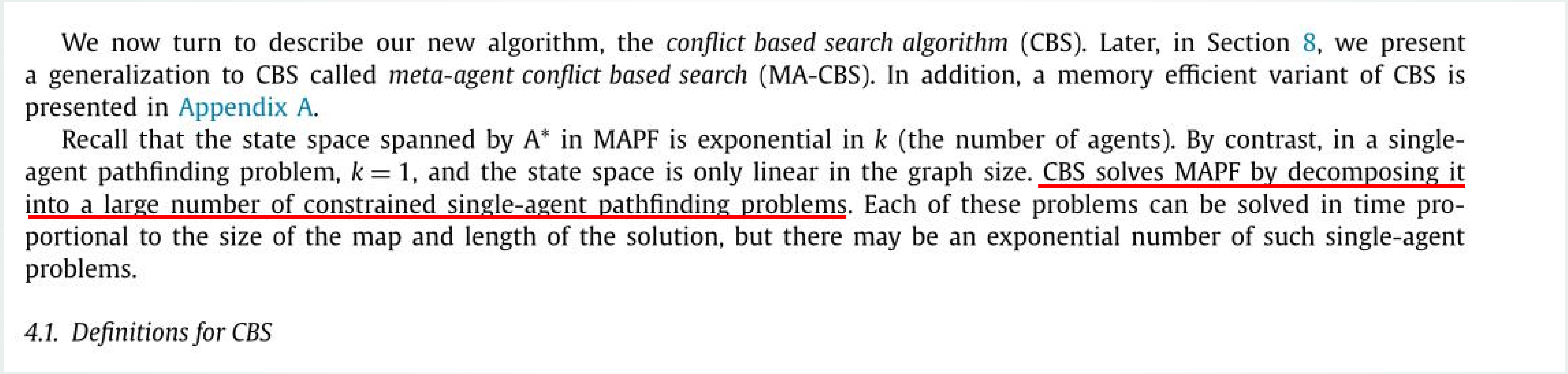
Constraint : 同一时刻一个点位最多只能被一个机器人占据

Types ： 集中式；分布式；混合式

Task： 得到由一系列无冲突路径集合组成的solution

Cost function: makespan, energy consumption, et. al.

1. CBS原理



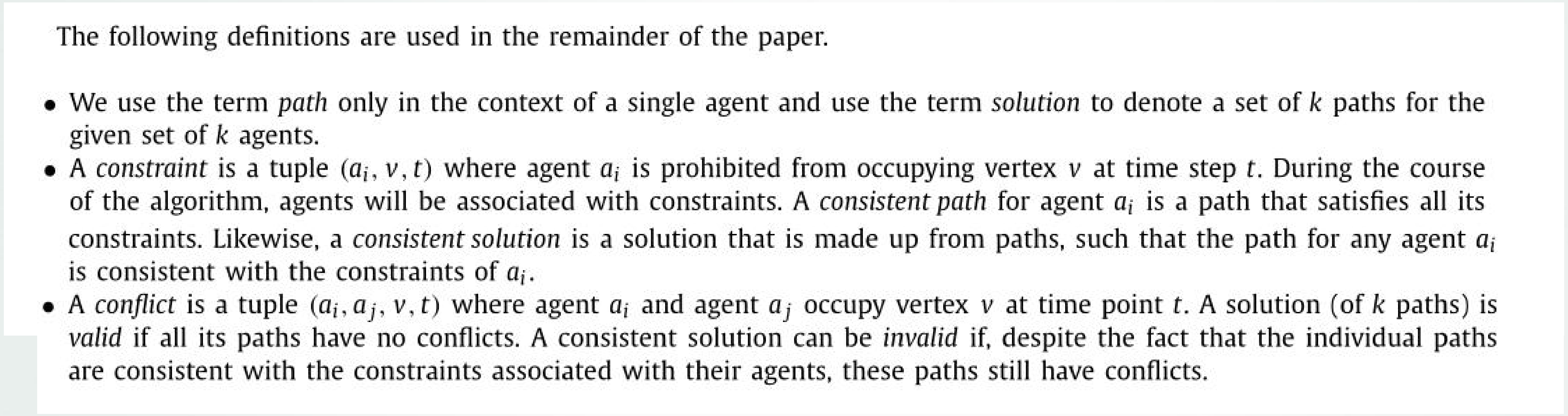
We now turn to describe our new algorithm, the *conflict based search algorithm* (CBS). Later, in Section 8, we present a generalization to CBS called *meta-agent conflict based search* (MA-CBS). In addition, a memory efficient variant of CBS is presented in Appendix A.

Recall that the state space spanned by A* in MAPF is exponential in k (the number of agents). By contrast, in a single-agent pathfinding problem, $k = 1$, and the state space is only linear in the graph size. CBS solves MAPF by decomposing it into a large number of constrained single-agent pathfinding problems. Each of these problems can be solved in time proportional to the size of the map and length of the solution, but there may be an exponential number of such single-agent problems.

4.1. Definitions for CBS

MAPF的A*跨越的状态空间在k(代理数量)中是指数的。相比之下，在单代理寻路问题中，k = 1，状态空间在图的大小上只是线性的。CBS通过将MAPF分解成大量受约束的单智能体寻路问题来解决该问题。

1. CBS原理


The following definitions are used in the remainder of the paper.

- We use the term *path* only in the context of a single agent and use the term *solution* to denote a set of k paths for the given set of k agents.
- A *constraint* is a tuple (a_i, v, t) where agent a_i is prohibited from occupying vertex v at time step t. During the course of the algorithm, agents will be associated with constraints. A *consistent path* for agent a_i is a path that satisfies all its constraints. Likewise, a *consistent solution* is a solution that is made up from paths, such that the path for any agent a_i is consistent with the constraints of a_i.
- A *conflict* is a tuple (a_i, a_j, v, t) where agent a_i and agent a_j occupy vertex v at time point t. A solution (of k paths) is *valid* if all its paths have no conflicts. A consistent solution can be *invalid* if, despite the fact that the individual paths are consistent with the constraints associated with their agents, these paths still have conflicts.

Path: 单个机器人的路径

Solution：所有机器人的可行路径集合

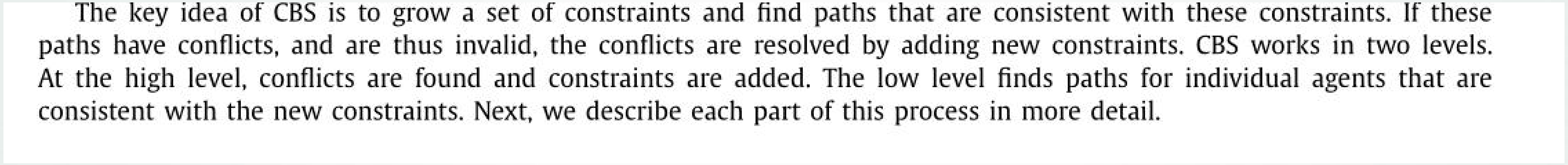
Constraint：（ai,v,t）表示机器人ai在t时刻禁止占用点位v

Consistent path：满足constraint（不违背constraint）的ai路径

Consistent solution：所有机器人均满足其对应constraint的可行路径组成的集合

Conflict：（ai, aj, v, t）表示ai和aj在t时刻同时占用点位v

1. CBS原理


The key idea of CBS is to grow a set of constraints and find paths that are consistent with these constraints. If these paths have conflicts, and are thus invalid, the conflicts are resolved by adding new constraints. CBS works in two levels. At the high level, conflicts are found and constraints are added. The low level finds paths for individual agents that are consistent with the new constraints. Next, we describe each part of this process in more detail.

CBS的关键思想是添加一系列的约束，并找到与这些约束一致的路径。如果这些路径有冲突，因而无效，则通过添加新的约束来解决冲突。

CBS分两个层次工作。在高层，发现冲突并增加约束。低级为各个代理找到符合新约束的路径。接下来，我们更详细地描述这个过程的每个部分。

讲到这里是否有些眼熟？是否与PP算法相似度很高？

双层优化？解耦方式？

1. CBS原理

CBS的双层结构： High level
Low level

HIGH LEVEL 包含五个重点内容

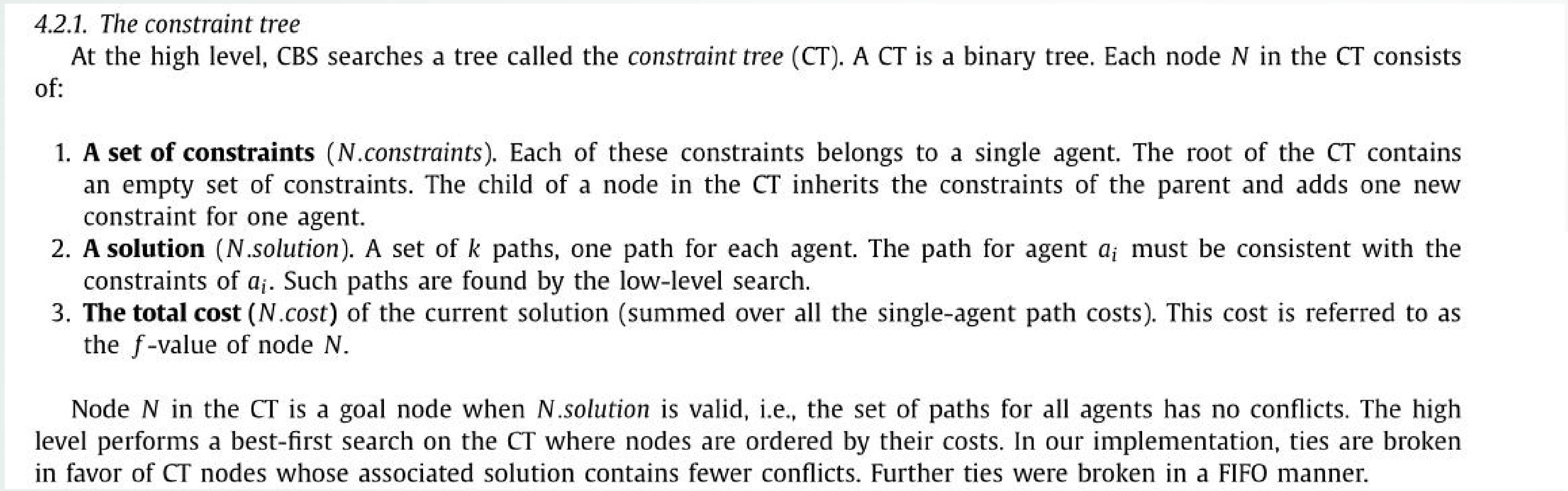
1. 冲突树(conflict tree, CT)
2. 处理CT中的node
3. 解决node处的conflict
4. 多机器人conflict
5. 边上的conflict

HIGH LEVEL 包含五个重点内容

1. **冲突树(conflict tree, CT)**
2. 处理CT中的node
3. 解决node处的conflict
4. 多机器人conflict
5. 边上的conflict

1. CBS原理

1. 冲突树(conflict tree, CT)



4.2.1. The constraint tree

At the high level, CBS searches a tree called the *constraint tree* (CT). A CT is a binary tree. Each node N in the CT consists of:

1. **A set of constraints** ($N.constraints$). Each of these constraints belongs to a single agent. The root of the CT contains an empty set of constraints. The child of a node in the CT inherits the constraints of the parent and adds one new constraint for one agent.
2. **A solution** ($N.solution$). A set of k paths, one path for each agent. The path for agent a_i must be consistent with the constraints of a_i. Such paths are found by the low-level search.
3. **The total cost** ($N.cost$) of the current solution (summed over all the single-agent path costs). This cost is referred to as the f-value of node N.

Node N in the CT is a goal node when $N.solution$ is valid, i.e., the set of paths for all agents has no conflicts. The high level performs a best-first search on the CT where nodes are ordered by their costs. In our implementation, ties are broken in favor of CT nodes whose associated solution contains fewer conflicts. Further ties were broken in a FIFO manner.

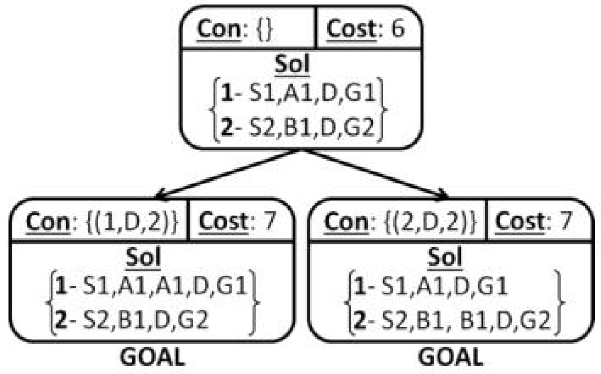


Fig. 4. An example of a *Constraint Tree* (CT).

重点
1. CT是一种二叉树，以N表示一个节点
2. 结点N中包含以下信息

a) 约束集，每个agent都有各自的约束；子节点继承父节点的所有约束，并加入新的约束；

b) 解（路径集合），每条路径对应一个agent，路径由low level搜索得到

c) 代价（成本），当前solution的代价值f，f的函数由设计者自定义。

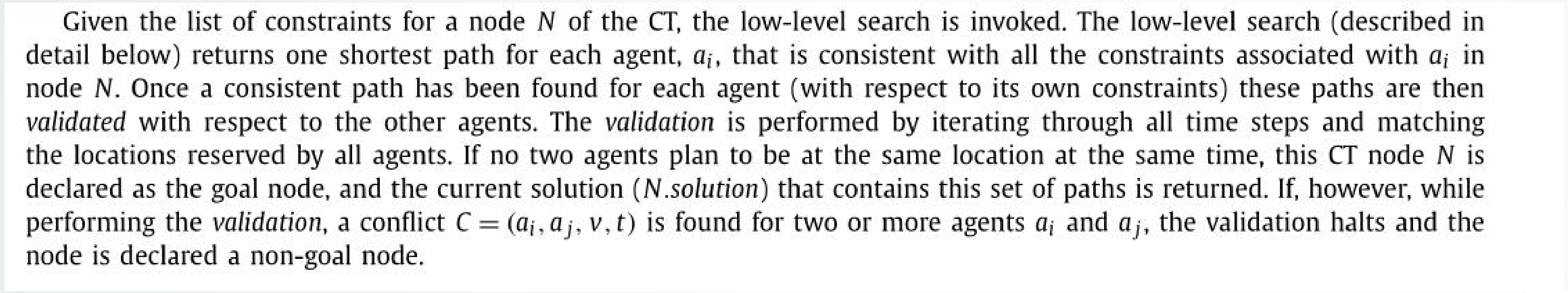
1. CBS原理

HIGH LEVEL 包含五个重点内容

1. 冲突树(conflict tree, CT)
2. **处理CT中的node**
3. 解决node处的conflict
4. 多机器人conflict
5. 边上的conflict

1. CBS原理

2. 处理CT中的node


Given the list of constraints for a node N of the CT, the low-level search is invoked. The low-level search (described in detail below) returns one shortest path for each agent, a_i, that is consistent with all the constraints associated with a_i in node N. Once a consistent path has been found for each agent (with respect to its own constraints) these paths are then *validated* with respect to the other agents. The *validation* is performed by iterating through all time steps and matching the locations reserved by all agents. If no two agents plan to be at the same location at the same time, this CT node N is declared as the goal node, and the current solution ($N.solution$) that contains this set of paths is returned. If, however, while performing the *validation*, a conflict $C = (a_i, a_j, v, t)$ is found for two or more agents a_i and a_j, the validation halts and the node is declared a non-goal node.

1.给定一个node N

2.确定N的约束集---

3.调用low level为每个agent搜索最短路径---

4.ai和aj路径比对---

5.如果所有agent路径都无conflict，确定N为goal node, 返回solution;

6.如果C=（ai, aj, v, t）, 确定N为non-goal node

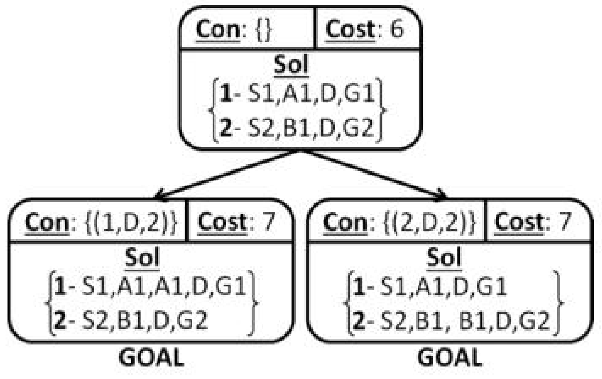


Fig. 4. An example of a *Constraint Tree* (CT).

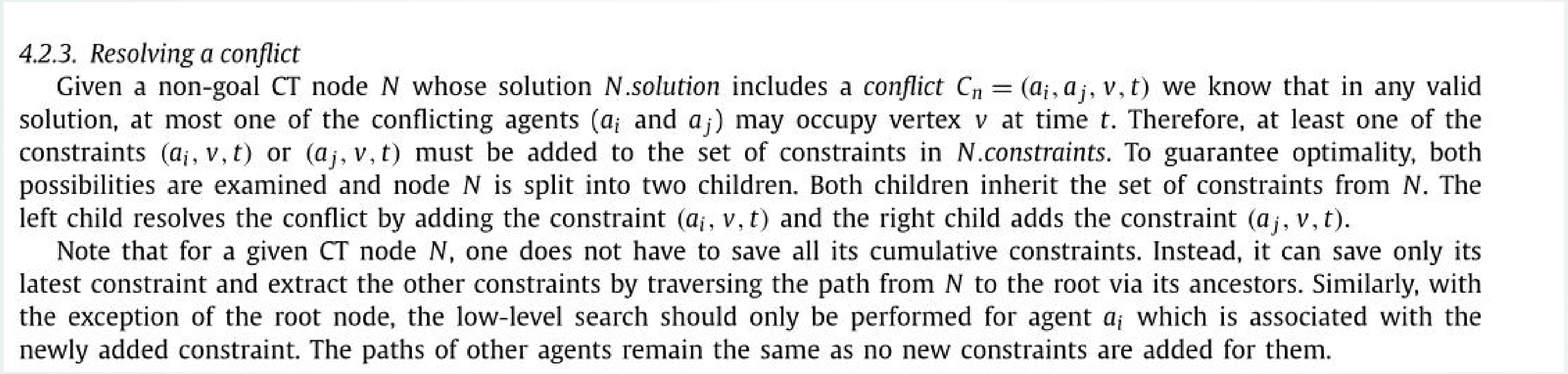
1. CBS原理

HIGH LEVEL 包含五个重点内容

1. 冲突树(conflict tree, CT)
2. 处理CT中的node
3. **解决node处的conflict**
4. 多机器人conflict
5. 边上的conflict

1. CBS原理

3. 解决node处的conflict



4.2.3. Resolving a conflict

Given a non-goal CT node N whose solution $N.solution$ includes a *conflict* $C_n = (a_i, a_j, v, t)$ we know that in any valid solution, at most one of the conflicting agents (a_i and a_j) may occupy vertex v at time t. Therefore, at least one of the constraints (a_i, v, t) or (a_j, v, t) must be added to the set of constraints in $N.constraints$. To guarantee optimality, both possibilities are examined and node N is split into two children. Both children inherit the set of constraints from N. The left child resolves the conflict by adding the constraint (a_i, v, t) and the right child adds the constraint (a_j, v, t).

Note that for a given CT node N, one does not have to save all its cumulative constraints. Instead, it can save only its latest constraint and extract the other constraints by traversing the path from N to the root via its ancestors. Similarly, with the exception of the root node, the low-level search should only be performed for agent a_i which is associated with the newly added constraint. The paths of other agents remain the same as no new constraints are added for them.

对于non-goal结点N，C=(ai, aj, v, t):

1. 冲突产生，至少有一个agent需要让步，也就是加入约束；
2. 随机选择一个agent加约束是可行的
3. 同时给冲突双方都加约束，可以提高效率，也就是（ai, v, t）以及（aj, v, t）
4. 加入约束后继续搜索，选择代价小的一方保留，另一方舍弃，也就是确定谁让谁，谁先谁后

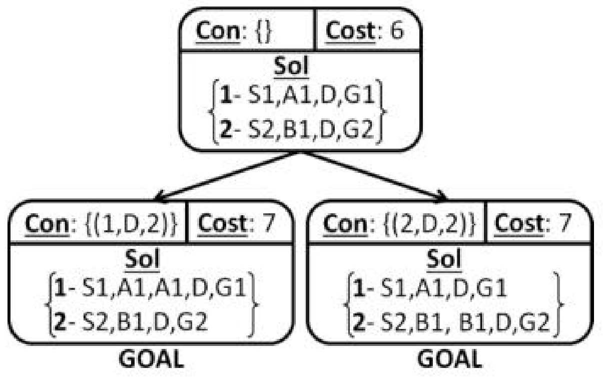


Fig. 4. An example of a *Constraint Tree* (CT).

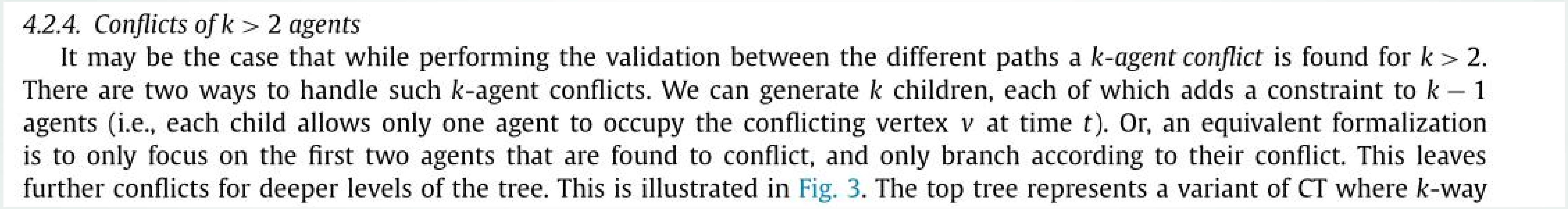
1. CBS原理

HIGH LEVEL 包含五个重点内容

1. 冲突树(conflict tree, CT)
2. 处理CT中的node
3. 解决node处的conflict
4. **多机器人conflict**
5. 边上的conflict

1. CBS原理

4. 多机器人conflict


4.2.4. Conflicts of k > 2 agents

It may be the case that while performing the validation between the different paths a *k-agent conflict* is found for $k > 2$. There are two ways to handle such k-agent conflicts. We can generate k children, each of which adds a constraint to $k - 1$ agents (i.e., each child allows only one agent to occupy the conflicting vertex v at time t). Or, an equivalent formalization is to only focus on the first two agents that are found to conflict, and only branch according to their conflict. This leaves further conflicts for deeper levels of the tree. This is illustrated in Fig. 3. The top tree represents a variant of CT where k-way

1. 针对三个及三个以上的agent冲突进行消除
2. 两种方式

   a) 与上面的冲突消除方式一样，假设有3个agent发生冲突，那么每个agent都对应产生一个子agent，加入相应的约束

   b) 只考虑前两个agent的冲突，剩余的agent不考虑

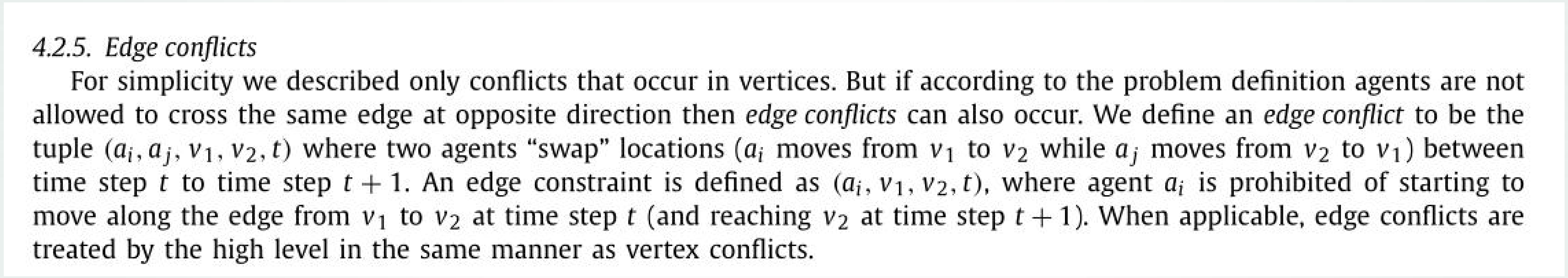
1. CBS原理

HIGH LEVEL 包含五个重点内容

1. 冲突树(conflict tree, CT)
2. 处理CT中的node
3. 解决node处的conflict
4. 多机器人conflict
5. **边上的conflict**

1. CBS原理

5. 边上的conflict



4.2.5. Edge conflicts

For simplicity we described only conflicts that occur in vertices. But if according to the problem definition agents are not allowed to cross the same edge at opposite direction then *edge conflicts* can also occur. We define an *edge conflict* to be the tuple (a_i, a_j, v_1, v_2, t) where two agents "swap" locations (a_i moves from v_1 to v_2 while a_j moves from v_2 to v_1) between time step t to time step $t + 1$. An edge constraint is defined as (a_i, v_1, v_2, t), where agent a_i is prohibited of starting to move along the edge from v_1 to v_2 at time step t (and reaching v_2 at time step $t + 1$). When applicable, edge conflicts are treated by the high level in the same manner as vertex conflicts.

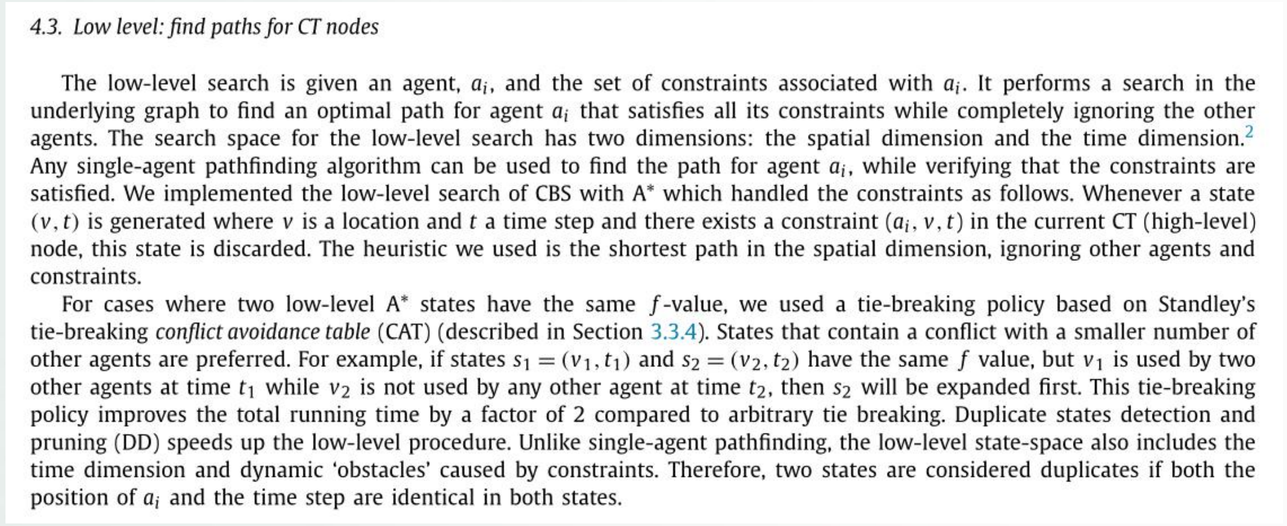
1.这一条考虑的是同一条路径（边）上可能存在的相向而行问题

2. C=(ai, aj, v1, v2, t)表示同一时刻ai要从v1到v2， aj要从v2到v1.

3. 解决办法是给ai添加约束（ai, v1, v2, t）,也就是说t时刻ai被禁止从v1到v2

1. CBS原理

CBS的双层结构： High level
Low level


4.3. Low level: find paths for CT nodes

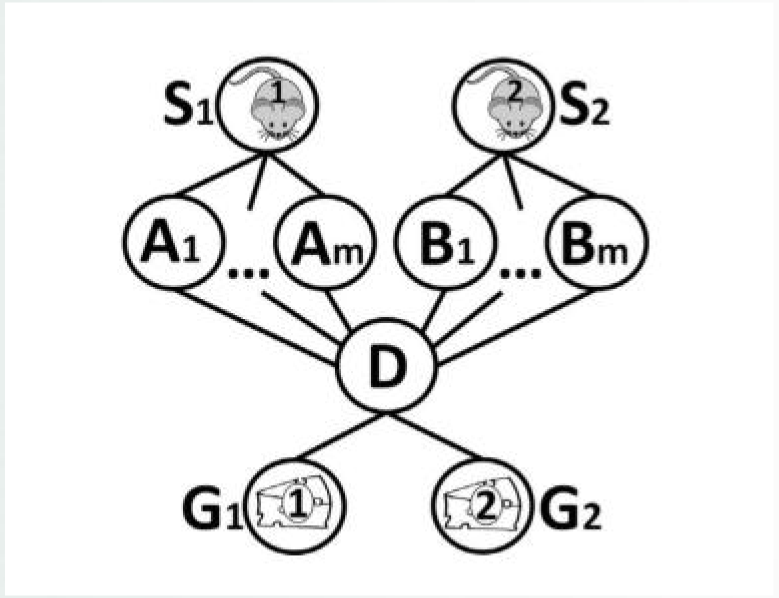
The low-level search is given an agent, a_i, and the set of constraints associated with a_i. It performs a search in the underlying graph to find an optimal path for agent a_i that satisfies all its constraints while completely ignoring the other agents. The search space for the low-level search has two dimensions: the spatial dimension and the time dimension.[2] Any single-agent pathfinding algorithm can be used to find the path for agent a_i, while verifying that the constraints are satisfied. We implemented the low-level search of CBS with A* which handled the constraints as follows. Whenever a state (v, t) is generated where v is a location and t a time step and there exists a constraint (a_i, v, t) in the current CT (high-level) node, this state is discarded. The heuristic we used is the shortest path in the spatial dimension, ignoring other agents and constraints.

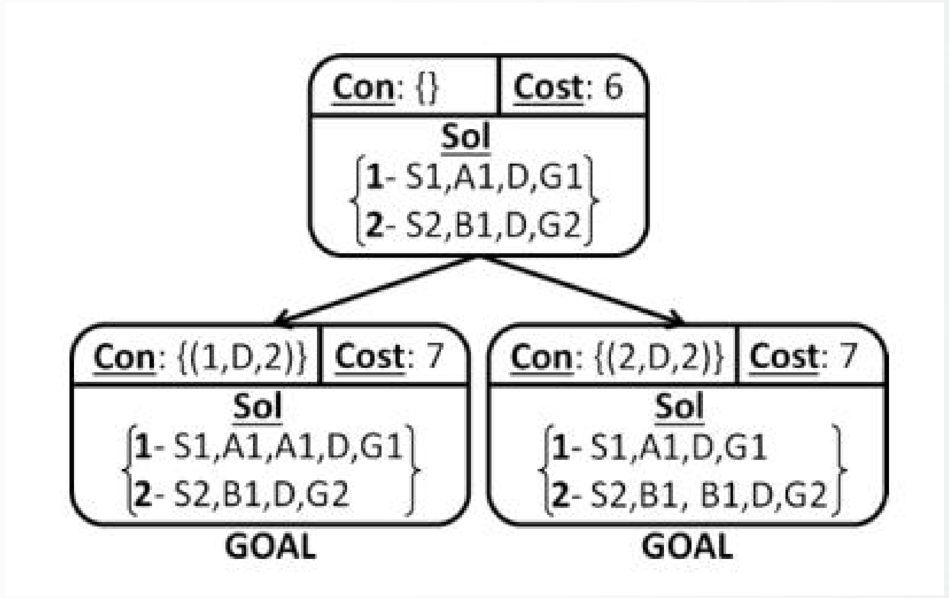
For cases where two low-level A* states have the same f-value, we used a tie-breaking policy based on Standley's tie-breaking *conflict avoidance table* (CAT) (described in Section 3.3.4). States that contain a conflict with a smaller number of other agents are preferred. For example, if states $s_1 = (v_1, t_1)$ and $s_2 = (v_2, t_2)$ have the same f value, but v_1 is used by two other agents at time t_1 while v_2 is not used by any other agent at time t_2, then s_2 will be expanded first. This tie-breaking policy improves the total running time by a factor of 2 compared to arbitrary tie breaking. Duplicate states detection and pruning (DD) speeds up the low-level procedure. Unlike single-agent pathfinding, the low-level state-space also includes the time dimension and dynamic 'obstacles' caused by constraints. Therefore, two states are considered duplicates if both the position of a_i and the time step are identical in both states.

重点：

1. Low level的搜索过程是：给定了ai以及ai的约束--- 搜索--- 找到最短路径pi
2. Low level 包含两个维度：时间，空间
3. 所有的单agent路径寻优算法都可以作为low level的搜索算法
4. 在这一层中，迭代和更新的原理是，对于当前状态S=（v, t），如果存在一个冲突（ai，v, t），那么S要被舍弃
5. 决胜机制：如果ai有两条代价相同的路径，s1=(v1, t1), s2=(v2, t2), f1=f2，采用决胜机制判定选择最短路径,具体判定方法是，如果t1时刻v1处被其他agent占据，而t2时刻v2处没有冲突，那么判定s2为最短路径的状态

1. CBS原理


S1
S2
A1
Am
B1
Bm
D
G1
G2

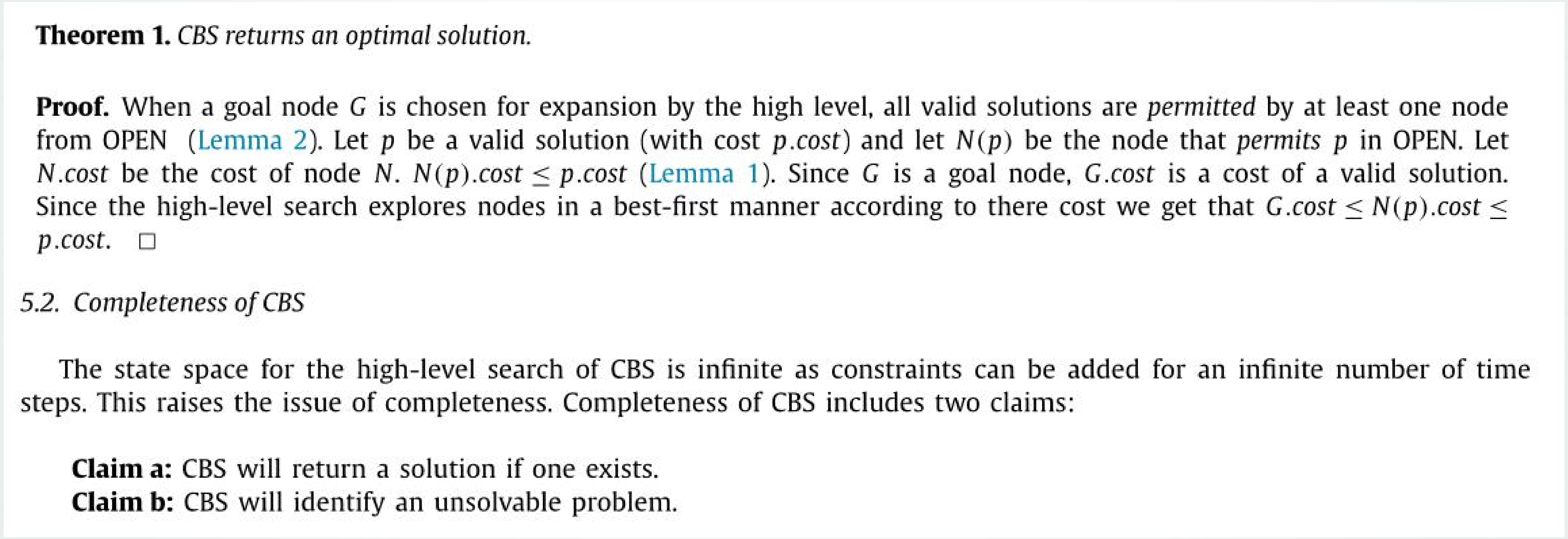

Con: {}
Cost: 6
Sol
1- S1,A1,D,G1
2- S2,B1,D,G2
Con: {(1,D,2)}
Cost: 7
Sol
1- S1,A1,A1,D,G1
2- S2,B1,D,G2
GOAL
Con: {(2,D,2)}
Cost: 7
Sol
1- S1,A1,D,G1
2- S2,B1, B1,D,G2
GOAL

2. CBS的优势与缺陷

CBS的特点：

1. 最优性
2. 完备性

缺点也隐含在了特点之中，最优性和完备性决定了CBS无法逃脱指数爆炸的怪圈



Theorem 1. *CBS returns an optimal solution.*

Proof. When a goal node G is chosen for expansion by the high level, all valid solutions are *permitted* by at least one node from OPEN (Lemma 2). Let p be a valid solution (with cost $p.cost$) and let $N(p)$ be the node that *permits* p in OPEN. Let $N.cost$ be the cost of node N. $N(p).cost \leq p.cost$ (Lemma 1). Since G is a goal node, $G.cost$ is a cost of a valid solution. Since the high-level search explores nodes in a best-first manner according to there cost we get that $G.cost \leq N(p).cost \leq p.cost$. □

5.2. Completeness of CBS

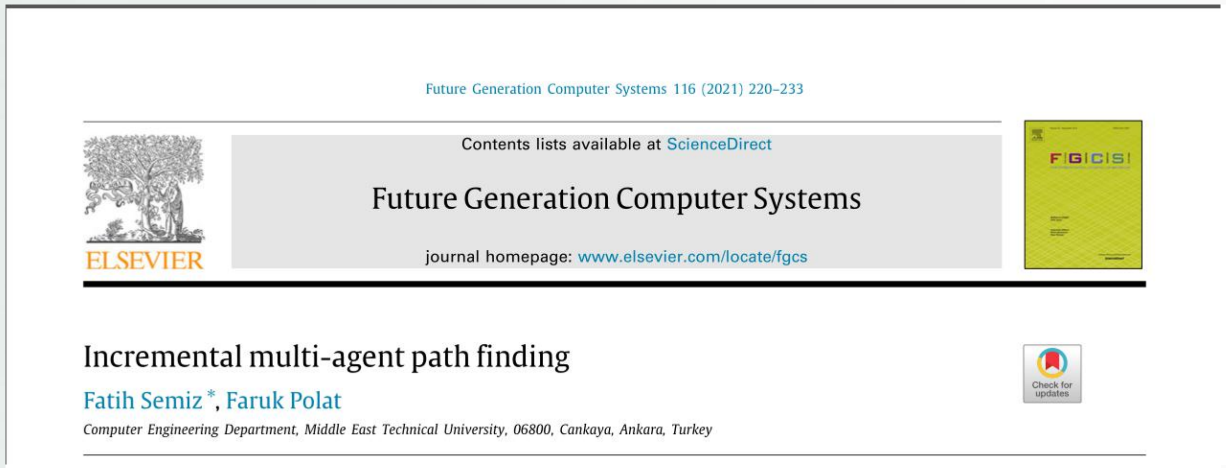
The state space for the high-level search of CBS is infinite as constraints can be added for an infinite number of time steps. This raises the issue of completeness. Completeness of CBS includes two claims:

Claim a: CBS will return a solution if one exists.
Claim b: CBS will identify an unsolvable problem.

3. 改进CBS的思路

1. 改进High level的约束
2. 更换Low level的算法
3. 改进low level的算法

3. 改进CBS的思路


Future Generation Computer Systems 116 (2021) 220–233

Contents lists available at ScienceDirect

Future Generation Computer Systems

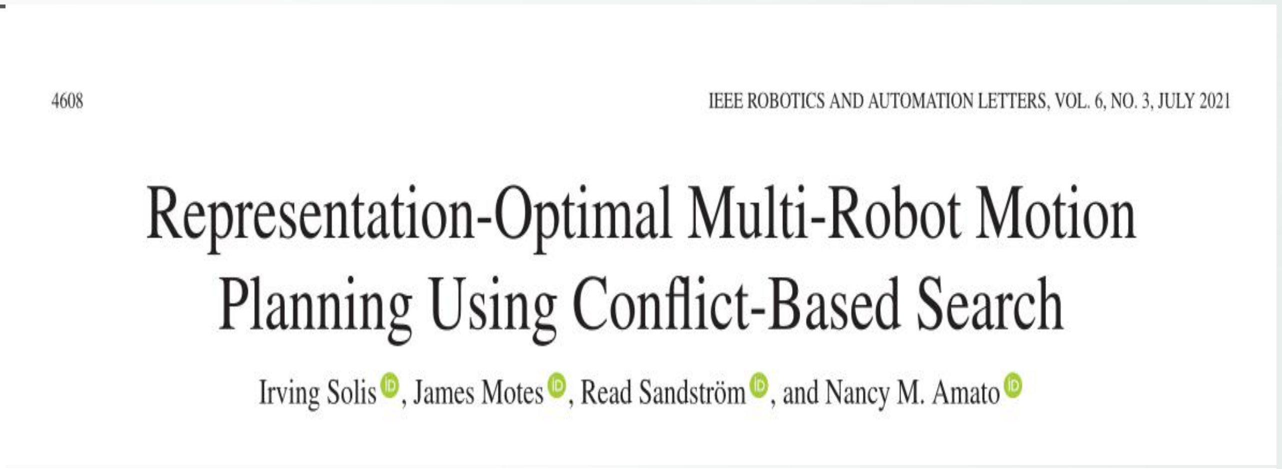
journal homepage: www.elsevier.com/locate/fgcs

ELSEVIER

Incremental multi-agent path finding

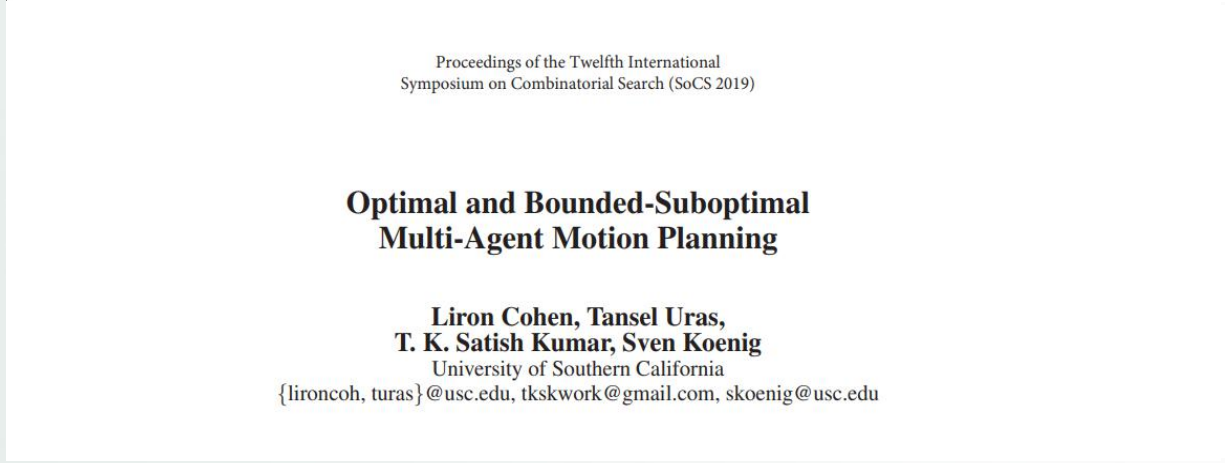
Fatih Semiz [*], Faruk Polat

Computer Engineering Department, Middle East Technical University, 06800, Cankaya, Ankara, Turkey


4608 IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 6, NO. 3, JULY 2021

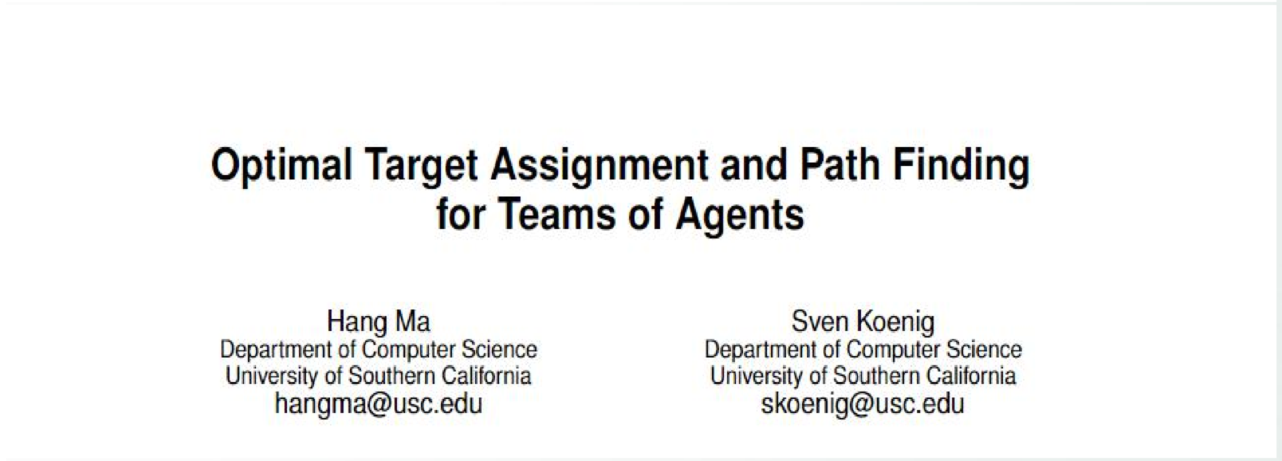
Representation-Optimal Multi-Robot Motion Planning Using Conflict-Based Search

Irving Solis, James Motes, Read Sandström, and Nancy M. Amato


Proceedings of the Twelfth International Symposium on Combinatorial Search (SoCS 2019)

Optimal and Bounded-Suboptimal Multi-Agent Motion Planning

Liron Cohen, Tansel Uras, T. K. Satish Kumar, Sven Koenig
University of Southern California
{lironcoh, turas}@usc.edu, tkskwork@gmail.com, skoenig@usc.edu


Optimal Target Assignment and Path Finding for Teams of Agents

Hang Ma
Department of Computer Science
University of Southern California
hangma@usc.edu

Sven Koenig
Department of Computer Science
University of Southern California
skoenig@usc.edu

https://www.bilibili.com/read/cv10556167

总结

1. CBS的思想较为简单，表面看是分为高层和底层两个层级，本质上还是单层算法
2. 之所以说本质上是单层算法，原因是高层次的作用只是提供约束，其本质就是设计规则，更合理的说法应该是基于启发式规则的MAPF算法
3. CBS的流程是，高层次检测conflict，如有conflict则添加约束；低层次执行搜索任务。
4. 与PP算法相似，CBS也是一种利用解耦降低复杂度的方法，此外，高层次的约束方式与PP算法的优先级约束以及附加的交通规则类似，都是通过添加启发式规则来提高算法性能

作业： 设计算法求解迷宫寻路问题

要求：

1. 迷宫规模50X50，障碍物可自行设置
2. Agent数量分别设置为2，5，10进行三组测试
3. 对比CBS算法与A*算法（其他算法也可）的性能