Advances in Applied Mathematics 应用数学进展, 2022, 11(4), 1857-1869
Published Online April 2022 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2022.114203

Hans汉斯

动态有向超图中限制不交B-路算法设计

米文燕，张淑蓉

太原理工大学，数学学院，山西 太原

收稿日期：2022年3月20日；录用日期：2022年4月14日；发布日期：2022年4月22日

摘　要

超图在现实生活中有很重要的应用价值，比如信息传递、货物运输、商品配送等问题都可以归约到超图中建立数学模型并设计优化算法。而网络环境是会随时间发生连续动态变化的，故本文主要研究动态超图中的连通性问题。同时，由于大规模网络中故障的发生是不可避免的，而且是极具破坏性的，所以，提高网络的生存性能，保证网络的容错性有很重要的研究价值。设计不交超路径是提高网络容错性的主要解决方案。由于超路中B-路有很好的结构性质和广泛的应用背景，因此，本文在时变超图网络中考虑满足时间限制的不交B-路构建问题。目前由于动态网络研究的复杂性，连续时间动态网络背景的处理方法大多是采用时间离散化转换为静态网络去求近似解，本文考虑当给定起始时刻时，在时间范围$[0,T]$内每条超弧的延迟函数为连续时间动态函数的情况下，针对不交B-路问题给出最优解的求解算法，并证明算法的正确性及运算复杂度。

关键词

有向超图，容错性，动态网络，B-路，不交路径

Design of Restricted Disjoint B-Path Algorithm in Dynamic Directed Hypergraph

Wenyan Mi, Shurong Zhang

College of Mathematics, Taiyuan University of Technology, Taiyuan Shanxi

Received: Mar. 20th, 2022; accepted: Apr. 14th, 2022; published: Apr. 22nd, 2022

Abstract

Hypergraph has very important application value in real life, for example, information transmis-

文章引用: 米文燕, 张淑蓉. 动态有向超图中限制不交 B-路算法设计[J]. 应用数学进展, 2022, 11(4): 1857-1869.
DOI: 10.12677/aam.2022.114203

sion, goods transportation, commodity distribution and other problems can be reduced to hypergraph to establish mathematical model and design optimization algorithm. The network environment will change continuously and dynamically with time, so this paper mainly studies the connectivity problem in dynamic hypergraph. At the same time, the occurrence of faults in large-scale networks is inevitable and extremely destructive. Therefore, it has important research value to improve the survival performance of networks and ensure the fault tolerance of networks. Designing disjoint hyperpath is the main solution to improve network fault tolerance. Because the B-path in hyperpath has good structural properties and wide application background, we consider the construction of disjoint B-paths satisfying time constraints in time-varying hypergraph networks in this paper. At present, due to the complexity of dynamic network research, most of the processing methods of continuous time dynamic network background adopt time discretization to static network to obtain approximate solutions. However, in this paper, when the starting time is given, the delay function of each super-arc within the time range $[0,T]$ is continuous time dynamic function, the algorithm for solving the optimal solution of the disjoint B-path problem is given. Finally, the correctness and computational complexity of the algorithm are proved.

Keywords

Directed Hypergraph, Fault Tolerance, Dynamic Network, B-Path, Disjoint Path

1. 引言

超图作为一般图的推广，将点集合视为超边的结构特点使其在网络中的应用可以有效反应多层次的节点关系并得到广泛研究[1] [2] [3]。特别是有向超图涉及多个学科领域，如人工智能[4] [5]、数据库[6]、运筹学[7] [8]等。同时，在许多现实问题中，不同时刻下的运输成本或运输量、信息的传输过程，都会有所不同，会随时间发生连续变化，因此，研究动态超图中的路由设计是亟待解决的问题之一。在动态网络中，通信网络节点或链路在信息传输时因受到干扰或攻击发生故障，会对网络功能产生影响，为了提高网络容错性，在主功能路径的基础上设计备份路径来传输信息是有效的解决方案。因此，在动态超网络背景下设计动态不交路是重要的研究课题。

超图中 B-路具有很好的结构性质和应用背景。早在 1989 年，学者 Italiano 和 Nanni [9]证明了在超图中的最短 B-路问题是 NP-难问题。到 1993 年，Gallo 等学者[10]提出在有向超图中，当网络中的超弧满足权值函数递增且不存在负圈时最短 B-路问题是可以多项式时间内解决的，并给出最短 B-树算法。但现实网络中往往会存在故障风险，为保证服务连续性，最常见的解决方案是设计备份路径，保证网络的容错性。

为了提高网络的容错性能，已经有不少学者研究了一般图中静态不交路的问题。早在 1974 年，Suurballe [11]利用最短路径的标记方法给出了在静态网络中查找 k 条 min-sum 内部节点不相交路由算法。在 1984 年，Suurballe 等人[12]又提出在静态网络中查找两条 min-sum 边不相交路由算法。之后很多研究者先后研究了不交路 minsum-minmin [13] [14]、minsum-minmax [15]等优化问题。虽然关于静态网络中不交路问题的研究已经很成熟，但动态超网络中的相关研究还很少。

目前，由于动态网络结构的复杂性，大量文献的研究成果都采用了时间离散化的方法，将时变赋权函数(延迟，容量，能耗等)的时间范围进行离散化，把时变网络上的动态问题简化为静态问题。由此对问题进行简化并得到近似解[16] [17]，但近似度的提高需要以时间细化程度为代价，增加了运算时间的复杂性。因此为了设计有效算法并获得最优解，一些学者对连续时间动态网络问题进行了研究。比如，在解决与时间相关的最短路径(TDSP)问题上，Ding [18]等人在先进先出(FIFO)连续时变网络中寻找 TDSP 问题的最优解，并提出时间复杂度为$O\left(\left(n\log n+m\right)\alpha\left(T\right)\right)$的有效算法，在给定时间区间$\left[0,T\right]$内$\alpha\left(T\right)$个子时间段中均得到了最优路径。Wang 等人[19]采用了一种层次图划分的方法解决 TDSP 问题，并给出时间复杂度为$O\left(\log_2^2\cdot k\cdot n\cdot\log_2^2\alpha\left(T\right)\right)$的算法，其中$k$是图划分得到的最小子图的个数。由于超图结构关系较复杂，在超图背景下与动态路径设计的相关文献还很缺乏。本文在一般超图$V_\Pi=\cup_{e_i\in E_\Pi}e_i$，$V_\Pi\subseteq V$相关工作的基础上研究动态超图中的限制不交 B-路问题，即在动态网络中，当时间范围是$\left[0,T\right]$且在某一固定时刻出发时从源点到汇点的两条内部不交 B-路。本文主要在于保证网络中两条路径的节点和超弧均是动态不交的，并且均在$\left[0,T\right]$内可以到达目标节点。

本文结构如下：第二节阐述了超图和动态超图中的基本概念；第三节分为两部分，第一部分提出在连续时间动态超网络中找一条$\left[0,T\right]$时间范围内的限制动态 B-路算法，第二部分研究动态超图网络中的限制不交 B-路并给出有效算法，同时在本节中对算法进行实例说明，证明了算法的正确性并计算了时间复杂度，之后，将动态不交 B-路算法推广到动态不交 F-路算法；第四部分总结全文。

2. 模型建立

2.1. 超图

定义 1 (**超图**[10]) 一个超图$\mathscr{H}=\left(V,E\right)$是一个有序对，其中$V=\left\{v_1,v_2,\cdots,v_{n-1},v_n\right\}$表示节点集，$E=\left\{e_1,e_2,\cdots,e_m\right\}$表示超弧集，并且对于$\forall i=1,2,\cdots,m$，有$e_i\subseteq V$。特别的，当$\left|e_i\right|=2$时，超图就是一般图。超图$\mathscr{H}$的大小定义为超弧基数的和：

$$size\left(\mathscr{H}\right)=\sum_{e_i\in E}\left|e_i\right|$$

一个有向的超弧e是一个有序对，$e=\left(T\left(e\right),H\left(e\right)\right)$，$T\left(e\right)$和$H\left(e\right)$分别为弧$e$的尾节点和头节点集合，$T\left(e\right)\cap H\left(e\right)=\phi$。当$\left|H\left(e\right)\right|=1$时，超弧$e$被称为 B-弧。设$FS\left(v\right)$为尾节点包含节点$v$的超弧，$FS\left(v\right)=\left\{e\in E:v\in T\left(e\right)\right\}$，$BS\left(v\right)$为头节点包含节点$v$的超弧，$BS\left(v\right)=\left\{e\in E:v\in H\left(e\right)\right\}$。由有向超弧组成的超图称为有向超图。下文中，为简便，将有向超图简写为超图。

定义 2 (**简单路** ***P*** [10]) 给定超图$\mathscr{H}\left(V,E\right)$，其中$V=\left\{v_1,v_2,\cdots,v_{n-1},v_n\right\}$，$E=\left\{e_1,e_2,\cdots,e_m\right\}$。一条简单路$P$是指一个有限非空序列$P=\left\{v_1=v_s,e_{i_1},v_2,e_{i_2},v_3,\cdots,e_{i_{k-1}},v_k=v_d\right\}$，它的项交替的为节点和超弧，并且节点和超弧都互不相同。其中源点$v_s\in T\left(e_{i_1}\right)$，目标节点$v_d\in H\left(e_{i_{k-1}}\right)$，且$v_j\in H\left(e_{i_{j-1}}\right)\cap T\left(e_{i_j}\right)$，$j=2,\cdots,k-1$。

接下来，给出 B-路的定义。

定义 3 (**B-路**[10]) 给定超图$\mathscr{H}\left(V,E\right)$，源点和目标节点分别为$v_s$和$v_d$。B-路$\Pi_{v_sv_d}$是一个极小的超图$\mathscr{H}_{\Pi_{v_sv_d}}=\left(V_\Pi,E_\Pi\right)$且满足以下条件：

1) $E_\Pi\subseteq E$；

2) $v_s,v_d\in V_\Pi$，其中$V_\Pi=\cup_{e_i\in E_\Pi}e_i,V_\Pi\in V$；

3) 对任一点$v_x\in V_\Pi$，v_x与源点v_s是连通的，也就是说，可以找到从v_s出发到达v_x有一条有向无环路。

在有了 B-路的定义后，实现了在复杂的超图网络中找一条路。接下来，我们考虑动态超图网络。

2.2. 动态超图网络

定义 4 (连续时间动态超图网络) 一个连续时间动态超图网络是一个有向超图 $\mathscr{H}=\left(V\left(\mathscr{H}\right),E\left(\mathscr{H}\right),W,T,v_s,v_d\right)$，定义如下：

1) $V\left(\mathscr{H}\right)$ 为图 $\mathscr{H}$ 的节点集，$\left|V\left(\mathscr{H}\right)\right|=n$。

2) $E\left(\mathscr{H}\right)$ 为图 $\mathscr{H}$ 的超弧集，对任一条有向超弧 $e=\left(T\left(e\right),H\left(e\right)\right)$，$T\left(e\right)$ 和 $H\left(e\right)$ 分别为弧 e 的尾节点和头节点集合，$T\left(e\right)\cap H\left(e\right)=\phi$。并且超弧 $e\subseteq V$，$\left|E(\mathscr{H})\right|=m$。

3) T 是本文考虑时间范围的上界。

4) $W=\left\{w_e\left(t\right),e\in E\left(\mathscr{H}\right)\right\}$，其 $w_e\left(t\right):\left[0,T\right]\mapsto R^+$ 表示超弧 e 的连续延迟函数，t 为超弧 e 的出发时间。对所有的 $t>T$，$w_e\left(t\right)=\infty$。

5) v_s,v_d 分别是源节点和目标节点。另外 $v_s\left(v_d\right)$ 不是任意一条超弧的头节点(尾节点)。

假设网络的拓扑结构和弧上的权值函数已确定。设 $FS\left(v\right)$ 为尾节点集中包含节点 v 的超弧，$FS\left(v\right)=\left\{e\in E\left(\mathscr{H}\right):v\in T\left(e\right)\right\}$，$BS\left(v\right)$ 为头节点集中包含节点 v 的超弧，$BS\left(v\right)=\left\{e\in E\left(\mathscr{H}\right):v\in H\left(e\right)\right\}$。本文中，每一个节点 v_i 的到达时间用 $f\left(i\right)$ 表示。特别的，由于每条超弧 e 的尾节点数 $\left|T\left(e\right)\right|\geq 1$，所以规定超弧 e 的出发时间 t 等于超弧尾节点 $T\left(e\right)$ 的最晚到达时间，故设每条超弧的出发时间 t 满足 $t=\max\left\{f\left(x\right)|v_x\in T\left(e\right)\right\}$，$f\left(x\right)$ 为节点 v_x 的到达时间。

在实际超图网络中，每条超弧 $e\in E\left(\mathscr{H}\right)$ 对应的延迟函数 $w_e\left(t\right)$ 是非负递增函数[10]。对于任意两个节点 $u\in T\left(e\right)$，$v\in H\left(e\right)$，从节点 u 到 v 存在一条有向超弧 e 当且仅当 $E\left(\mathscr{H}\right)$ 中有元素 $e=\left(T\left(e\right),H\left(e\right)\right)$。

定义 5 (节点时间对) 节点时间对 $\left(v_x,f\left(x\right)\right)$ 指的是在路径 P 上，到达节点 v_x 的时刻为 $f\left(x\right)$，其中 $v_x\in V\left(\mathscr{H}\right)$。

定义 6 (动态 B-路) 给定一个连续时间动态超图网络 $\mathscr{H}=\left(V\left(\mathscr{H}\right),E\left(\mathscr{H}\right),W,T\right)$，源节点为 v_s，目标节点为 v_d。设 $V_\Pi\left(t\right)$ 是动态 B-路的节点集，$E_\Pi\left(t\right)$ 是动态 B-路的有向超弧集，动态 B-路是一个极小的动态超图 $\mathscr{H}_\Pi\left(t\right)=\left(V_\Pi\left(t\right),E_\Pi\left(t\right)\right)$。一条动态 B-路 $\mathscr{H}_\Pi\left(t\right)$ 指的是当出发时间为 t 时，从源点 v_s 到目标节点 v_d 的一条超路，满足：

1) $E_\Pi\left(t\right)\subseteq E\left(\mathscr{H}\right)$。

2) $\left\{\left(v_s,t\right),\left(v_d,f\left(d\right)\right)\right\}\in V_\Pi\left(t\right)$，其中 $V_\Pi\left(t\right)=\cup_{e_i\in E_\Pi\left(t\right)}e_i$，$V_\Pi\left(t\right)\subseteq V\left(\mathscr{H}\right)$。

3) 对 $\forall v_x\in V_\Pi\left(t\right)$，可以找到从源点 v_s 到 v_x 的一条有向无环路。

4) 经过每条超弧 e 的出发时间 t' 等于尾节点 $T\left(e\right)$ 的最大到达时间，$t'=\max\left\{f\left(x\right)|v_x\in T\left(e\right)\right\}$，即经过一条超弧的必要条件是在该超弧所有尾节点都到达的情况下才能出发。

在给出以上定义后，现在我们定义当出发时间为 t 时，在动态 B-路 $\mathscr{H}_{\Pi_{e_1e_m}}\left(t\right)$ 上从节点 $v_s\in T\left(e_1\right)$ 到达节点 $v_d\in H\left(e_m\right)$ 的旅行时间函数 $w_{\Pi_{e_1e_m}}\left(t\right)$：

$$w_{\Pi_{e_1e_m}}\left(t\right)=w_{\Pi_{e_1e_{m-1}}}\left(t\right)+w_{e_m}\left(t+w_{\Pi_{e_1e_{m-1}}}\left(t\right)\right)$$

$w_{\Pi_{e_1e_m}}\left(t\right)$ 的值表示从超弧 e_1 到超弧 e_{m-1} 的旅行时间加上超弧 e_m 的延迟。

从上述定义可知，B-路的到达时间为 $f\left(d\right)$。一条动态限制 B-路是指一条动态 B-路在时间范围 $\left[0,T\right]$ 内可以到达目标节点 v_d，即满足 $f\left(d\right)<T$。接下来给出在连续时间动态超图网络中限制不交 B-路的定义。

定义 7 (动态限制 *k* 不交 B-路) 给定一个连续时间动态超图网络 $\mathscr{H}=\left(V\left(\mathscr{H}\right),E\left(\mathscr{H}\right),W,T\right)$，节点 $v_s,v_d\in V\left(\mathscr{H}\right)$ 分别为源节点和目标节点，出发时间为 t，设 $V_{\Pi_i}\left(t\right),E_{\Pi_i}\left(t\right)$ 分别为第 i 条 B-路的节点集和超弧集。动态限制 k 不交 B-路是在时间范围 $\left[0,T\right]$ 内寻找 k 条 B-路 $\mathscr{H}_{\Pi_1}\left(t\right),\mathscr{H}_{\Pi_2}\left(t\right),\cdots,\mathscr{H}_{\Pi_k}\left(t\right)$，满足 $\cap_{i=1}^{k}V_{\Pi_i}\left(t\right)=\phi$，$\cap_{i=1}^{k}E_{\Pi_i}\left(t\right)=\phi$。

针对动态 B-路的设计，提出的优化问题如下：

问题 1 给定一个连续时间动态超图网络 $\mathscr{H}=\left(V\left(\mathscr{H}\right),E\left(\mathscr{H}\right),W,T\right)$，源点为 v_s，目标节点为 v_d，$W=\left\{w_e\left(t\right),e\in E\left(\mathscr{H}\right)\right\}$ 为超弧的连续时间延迟函数，出发时间为 t_s。在时间范围 $\left[0,T\right]$ 内，找一条动态限制 B-路 $\mathscr{H}_\Pi^*\left(t_s\right)=\left(V_\Pi^*\left(t_s\right),E_\Pi^*\left(t_s\right)\right)$。

问题 2 给定一个连续时间动态超图网络 $\mathscr{H}=\left(V\left(\mathscr{H}\right),E\left(\mathscr{H}\right),W,T\right)$，源点为 v_s，目标节点为 v_d，出发时间为 t_s。在时间范围 $\left[0,T\right]$ 内，找两条动态限制不交 B-路 $\mathscr{H}_{\Pi_1}\left(t_s\right)=\left(V_{\Pi_1}\left(t_s\right),E_{\Pi_1}\left(t_s\right)\right)$，$\mathscr{H}_{\Pi_2}\left(t_s\right)=\left(V_{\Pi_2}\left(t_s\right),E_{\Pi_2}\left(t_s\right)\right)$。

本文中，假设动态超图网络 $\mathscr{H}$ 满足 FIFO 性质，即对任意边 e 及边 e 的出发时间 t,t'，如果 $t<t'$，有 $t+w_e\left(t\right)\le t'+w_e\left(t'\right)$。并且假设动态超图网络 $\mathscr{H}$ 中是不存在负圈的。

3. 算法设计

在静态网络中，找不交路最初的方法是先找一条路，然后将这条路经过的节点和弧全部删去，再找另外一条路。但会存在一中情况，就是当删去第一条路之后，会影响到第二条路的构建，而原网络中是存在两条不交路的。这时就出现了“陷阱”问题。针对这种情况，有效的解决办法是在给定一条路的基础上去构造两条不交路[11]，而不是直接删除该条路。这样，算法可以同时找到两条不交路，而且解决了所谓的“陷阱”问题。在给定连续时间 $\left[0,T\right]$ 内，考虑时变的延迟函数，我们结合这种方法在连续动态超图网络中找两条动态限制不交 B-路。

3.1. 动态限制 B-路算法

在解决找两条动态限制不交 B-路问题之前，首先需要在动态超图网络中找到一条动态限制 B-路 $\mathscr{H}_\Pi^*\left(t_s\right)=\left(V_\Pi^*\left(t_s\right),E_\Pi^*\left(t_s\right)\right)$，其中，$V_\Pi^*\left(t_s\right)$ 为 B-路的节点集，$E_\Pi^*\left(t_s\right)$ 为 B-路的超弧集，t_s 为出发时间。结合 B-树的构造方法[10]，接下来，我们给出动态限制 B-路算法的求解过程。

给定一个动态超图网络 $\mathscr{H}=\left(V\left(\mathscr{H}\right),E\left(\mathscr{H}\right),W,T\right)$，集合 Q 中包含算法每次迭代需要遍历的节点集，集合 L 中包含已经遍历过的节点集。对于超图中的每条超弧 $e_j\in E\left(\mathscr{H}\right)$，$T\left(e_j\right)$ 为超弧 e_j 的尾节点集，设 k_j 为从集合 Q 删去的 $T\left(e_j\right)$ 中的节点数，即算法遍历过弧 e_j 的尾节点数。

在算法过程中，t_s 表示出发时间，节点 $v_s,v_d\in V\left(\mathscr{H}\right)$ 分别为源节点和目标节点。v_i 表示每次从集合 Q 中取的需要遍历的节点，当遍历的尾节点数 $k_j=\left|T\left(e_j\right)\right|$ 时，选取尾节点中到达时间最大的 t 作为超弧 e_j 的出发时间，$t=\max\left\{f\left(x\right)\mid v_x\in T\left(e_j\right)\right\}$。算法第 10 行中，对于属于 $H\left(e_j\right)$ 的节点 v_y 且 $v_y\notin L$，求得节点 v_y 的到达时间 $f\left(y\right)=w_{e_j}\left(t\right)+t$，并将节点 v_y 放入集合 L 中，表示该节点已经被遍历过，在下次找超弧头节点时不在查找这个节点，而是继续向后查询。

在满足到达时间 $f\left(y\right)<T$ 的情况下，进行算法第 13 行，标记 $p_v\left(y,f\left(y\right)\right)=e_j$ 以记录超路的信息。之后，判断该节点是否被遍历过(第 17 行)，如果没有，放到集合 Q 中以便下一次迭代。直到查找到目标节点 v_d 时跳出循环(第 14 行)，求出一条动态限制 B-路 $\mathscr{H}_\Pi^*\left(t_s\right)$。

算法 1：动态限制 B-路

1. Input: 一个动态超图网络 $\mathscr{H}\left(V\left(\mathscr{H}\right),E\left(\mathscr{H}\right),W,T,v_s,v_d\right)$，$v_s$ 点的出发时间 t_s；
2. Output: 一条动态限制 B-路 $\mathscr{H}_\Pi^*\left(t_s\right)=\left(V_\Pi^*\left(t_s\right),E_\Pi^*\left(t_s\right)\right)$；
3. Initialize: $f\left(i\right)=\infty$，$v_i\in V\left(\mathscr{H}\setminus\left\{v_s\right\}\right)$，对于每一条超弧 e_j，有 $k_j=0$，$f\left(s\right)=t_s$，$Q=L=\left\{v_s\right\}$；
4. While $Q\neq\phi$ do

Continued

5. select v_i from Q; $Q=Q-\{v_i\}$;
6. For each $e_j \in FS(v_i)$ do
7. $k_j = k_j + 1$;
8. If $k_j = |T(e_j)|$ then
9. $t = \max\{f(x) \mid v_x \in T(e_j)\}$;
10. For each $v_y \in H(e_j)$ and $v_y \notin L$ do
11. $f(y) = w_{e_j}(t) + t$; $L = L \cup \{v_y\}$;
12. If $f(y) < T$ then
13. $p_v(y, f(y)) = e_j$;
14. If $v_y = v_d$ then
15. return B-路 $\mathscr{H}_{\Pi}^{*}(t_s)$;
16. EndIf
17. If $v_y \notin Q$ then
18. $Q = Q \cup \{v_y\}$;
19. EndIf
20. Else
21. return step 5;
22. EndIf
23. EndFor
24. EndIf
25. EndFor
26. EndWhile

定理 1 算法 1 的复杂度为 $O(n \cdot size(\mathscr{H}))$。

证明：算法 1 第 5 行，在 Q 集合中节点的个数最多有 n 个，第 6 行对于节点 v_i 出去的弧 e_j，需要花费 $O(|T(e_j)|)$ 的运算时间，算法第 10 行中探寻弧 e 的头节点需要花费 $O(|H(e_j)|)$ 的运算时间，故算法 1 的复杂度为 $O(n \cdot size(\mathscr{H}))$。

当算法 1 结束后，得出一条动态限制 B-路 $\mathscr{H}_{\Pi}^{*}(t_s)$。接下来通过这条已知的 B-路来找两条动态限制不交 B-路。

3.2. 动态限制不交 B-路算法

本节的目标是在动态超图网络 $\mathscr{H}=(V(\mathscr{H}),E(\mathscr{H}),W,T)$ 中找两条动态限制不交的 B-路。为了避免陷阱问题，本文的算法思想是在已知一条限制 B-路 $\mathscr{H}_{\Pi}^{*}(t_s)$ 的基础上构造两条动态限制不交 B-路 $\mathscr{H}_{\Pi_1}(t_s)=(V_{\Pi_1}(t_s),E_{\Pi_1}(t_s))$，$\mathscr{H}_{\Pi_2}(t_s)=(V_{\Pi_2}(t_s),E_{\Pi_2}(t_s))$。

3.2.1. 准备工作

本问题面临的主要挑战是在动态超图中，算法找到的两条路 $\mathscr{H}_{\Pi_1}(t_s)$，$\mathscr{H}_{\Pi_2}(t_s)$ 需要满足经过的节点时间对是动态不交的，且两条路所有超弧的经过时间段是不交的。对于一条超弧 $e=(T(e),H(e))$，设 $f(T(e))$ 和 $f(H(e))$ 分别为超弧 e 头节点和尾节点的到达时间，即：

1) 对任意的节点时间对 $(v_i,f(i))\in V_{\Pi_1}(t_s)$，$(v_j,f(j))\in V_{\Pi_2}(t_s)$，有 $(v_i,f(i))\neq(v_j,f(j))$。

2) 对任意的超弧 $e_i \in E_{\Pi_1}(t_s)$，$e_j \in E_{\Pi_2}(t_s)$，超弧 e_i 经过的时间区间表示为 $I_1 = \left[f(T(e_i)), f(H(e_i)) \right]$，$e_j$ 经过的时间区间表示为 $I_2 = \left[f(T(e_j)), f(H(e_j)) \right]$，有 $I_1 \cap I_2 = \phi$。

故为保证以上两个条件，在找路的过程中需要避开已知路 $\mathscr{H}_{\Pi}^{*}(t_s)$ 中超弧经过的时间段，我们更新动态超图网络 $\mathscr{H}$ 中超弧 e 的延迟函数 $W' = \{w_e'(t), e \in E(\mathscr{H})\}$。

首先对已知动态限制 B-路 $\mathscr{H}_{\Pi}^{*}(t_s)$ 上所有超弧的延迟函数进行更新。对所有的 $e \in E_{\Pi}^{*}(t_s)$，假设弧 e 的尾节点集 $T(e)$ 中到达时间 $f(T(e))$ 最小的记为 a，即 $a = \min\{ f(T(e)) \mid e \in E_{\Pi}^{*}(t_s) \}$。头节点集 $H(e)$ 中到达时间 $f(H(e))$ 最大的记为 t''，即 $t'' = \max\{ f(H(e)) \mid e \in E_{\Pi}^{*}(t_s) \}$。故有：

$$t' = \arg\min_{t \in [0,T]} \{ a \le t + w_e(t) \le t'' \}$$

所以，在找两条路的过程中，对于每条在路 $\mathscr{H}_{\Pi}^{*}(t_s)$ 上的超弧，更新在时间段 $[t', t'')$ 的延迟函数。

$$w_e'(t) = \begin{cases} \infty, & t \in [t', t'') \\ w_e(t), & \text{otherwise.} \end{cases}$$

对于动态超图网络 $\mathscr{H}$ 中除 $\mathscr{H}_{\Pi}^{*}(t_s)$ 之外的其他超弧，他们的延迟函数保持不变。即：$w_e'(t) = w_e(t)$，$e \in \left(E(\mathscr{H}) - E_{\Pi}^{*}(t_s) \right)$。

这样，当我们在找两条不交路时，能避开 $\mathscr{H}_{\Pi}^{*}(t_s)$ 超弧走过的相应时间段，使找到的超弧满足动态边不交的条件。接下来给出实例说明。

例 1 假设弧 e 为动态限制 B-路 $\mathscr{H}_{\Pi}^{*}(t_s)$ 的一条超弧，弧 e 的尾节点集 $\{(v_1, 2), (v_2, 2.5)\}$，头节点集 $\{(v_3, 3.5)\}$，延迟函数 $w_e(t)$ 见图 1。根据以上步骤更新弧 e 的延迟函数 $w_e(t)$ 为 $w_e'(t)$，见图 2。

$$w_e(t) = \begin{cases} 1, & t \in [0, 3) \\ t - 2, & t \in [3, T]. \end{cases}$$

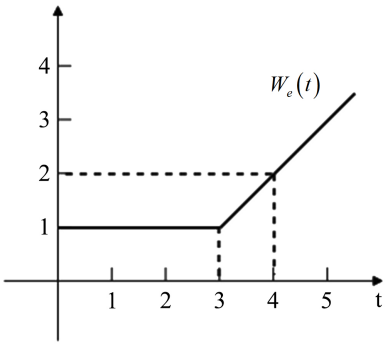

Figure 1. $w_e(t)$

图 1. $w_e(t)$

接下来我们求延迟函数 $w_e(t)$ 要更新的区间 $[t', t'')$。

$a = \min\{ f(1), f(2) \} = 2.$

$t'' = \max\{ f(3) \} = 3.5.$

$t' = \arg\min_{t \in [0,T]} \{ 2 \le t + w_e(t) \le 3.5 \} = 1.$

$$w_e'(t) = \begin{cases} 1, & t \in [0, 1); \\ \infty, & t \in [1, 3.5); \\ t - 2, & t \in [3.5, T]. \end{cases}$$

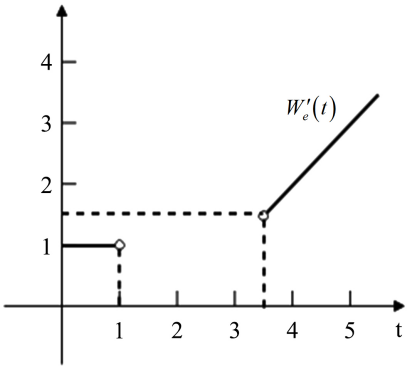

Figure 2. $w_e'(t)$

图 2. $w_e'(t)$

3.2.2. 算法设计与实现

在给出动态超图网络中求两条不交路的算法之前，我们先通过示例给出静态网络中基于一条路去找两条不交路的过程，并说明交错点的定义。

例 2 如图 3 所示，给定图 $\mathscr{H}(V,E)$，$V=\{v_1,v_2,v_3,v_4\}$，$E=\{e_1,e_2,e_3,e_4,e_5\}$。假设静态网络中每条弧 e 都有一个权值 w。节点 v_1 和 v_4 是源节点和目标节点。给出一条已知超路 $\mathscr{H}_\Pi^*=\left(\{v_1,v_2,v_3,v_4\},\{e_1,e_2,e_5\}\right)$，其中，超路节点集为 $V_\Pi^*=\{v_1,v_2,v_3,v_4\}$，弧集 $E_\Pi^*=\{e_1,e_2,e_5\}$。那么我们可以根据以下步骤获得两条不交路 $\mathscr{H}_{\Pi_1},\mathscr{H}_{\Pi_2}$。

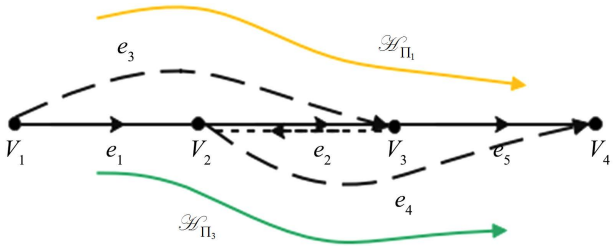

Figure 3. Hypergraph $\mathscr{H}(V,E)$

图 3. 超图 $\mathscr{H}(V,E)$

首先，我们可以从源节点 v_1 出发找到除 $\mathscr{H}_\Pi^*$ 之外的边 $e_3=(v_1,v_3)$，由于节点 $v_3\in V_\Pi^*$，记 v_3 为交错点。因此，为构造两条不交路，需要反向找节点 v_1 和 v_3 之间的节点 v_2。从 v_2 继续向后探查除 $\mathscr{H}_\Pi^*$ 之外的边 $e_4=(v_2,v_4)$。这样，就找到两条不交的路 $\mathscr{H}_{\Pi_1}=\left(\{v_1,v_3,v_4\},\{e_3,e_5\}\right)$，$\mathscr{H}_{\Pi_2}=\left(\{v_1,v_2,v_4\},\{e_1,e_4\}\right)$。

每次探查完交错点之后，我们将该交错点标记为 c 用于后续路径连接。交错点的个数记为 m。

接下来我们在动态超图网络 $\mathscr{H}=\left(V(\mathscr{H}),E(\mathscr{H}),W,T\right)$ 中构造不交路。

算法步骤如下：

1) 初始化。源点 v_s 的出发时间为 t_s，将源点出发经过的不属于 $V_\Pi^*(t_s)$ 的第一条超弧的头节点 $\left(v_i,f(i)\right)$ 放到候选点集 Q 中，初始化 $f(i)=t_s+w_{si}(t_s)$，$w_{si}(t_s)$ 为弧 $e=(v_s,v_i)$ 的延迟函数。集合 L 表示算法已经考虑过的节点集，初始化 $L=\{v_s\}$。参数 m 记录交错点的个数，$m=0$。c 标记交错点。

2) 求已知路 $\mathscr{H}_\Pi^*(t_s)$。通过算法 1，在动态超图网络 $\mathscr{H}=\left(V(\mathscr{H}),E(\mathscr{H}),W,T\right)$ 中求得一条动态限制 B-路 $\mathscr{H}_\Pi^*(t_s)=\left(V_\Pi^*(t_s),E_\Pi^*(t_s)\right)$，$V_\Pi^*(t_s)$ 和 $E_\Pi^*(t_s)$ 分别为路 $\mathscr{H}_\Pi^*(t_s)$ 经过的节点时间对集和超弧集。

3) 运行算法 2。首先，当 Q 不为空时，选择其中的候选点$\left(v_i, f\left(i\right)\right)$，然后开始找点 v_i 出去的超弧 $e_j \in FS\left(v_i\right)$，结合 B-弧的查找方法[10]，当遍历完超弧的所有尾节点后(算法第 9~10 行)，选择到达时间最大的尾节点，并将其到达时间作为超弧 e_j 的出发时间，接着当超弧的头节点 $v_y \notin L$ 时，求其到达时间 $f\left(y\right)$。判断 $f\left(y\right)$，当 $f\left(y\right)<T$ 时，说明弧 e_j 与 $\mathscr{H}_{\Pi}^{*}\left(t_s\right)$ 的弧是动态不交的，记录路信息并将头节点 v_y 放入集合 L 中，进行算法 3 判断交错点。否则，算法判断 Q 是否为空，如果不为空，说明可以继续重新探查下一个节点，返回算法第 4 步，如果为空，则说明超图中没有满足条件的路径。

算法 2：动态限制不交 B-路

1. Input: $\mathscr{H}\left(V\left(\mathscr{H}\right),E\left(\mathscr{H}\right),W',T,v_s,v_d\right)$，一条动态限制 B-路 $\mathscr{H}_{\Pi}^{*}\left(t_s\right)=\left(V_{\Pi}^{*}\left(t_s\right),E_{\Pi}^{*}\left(t_s\right)\right)$，$t\in\left[0,T\right]$，出发时间 t_s；
2. Output: 动态限制不交 B-路 $\mathscr{H}_{\Pi_1}\left(t_s\right)$，$\mathscr{H}_{\Pi_2}\left(t_s\right)$；
3. Initialize: $Q=\left\{\left(v_i, f\left(i\right)\right)\middle|\left(v_s,v_i\right)\in E\left(\mathscr{H}\right),\left(v_i, f\left(i\right)\right)\notin V_{\Pi}^{*}\left(t_s\right)\right\}$，其中 $f\left(i\right)=t_s+w_{si}\left(t_s\right)$；$V_{\Pi_1}\left(t_s\right)=\left\{\left(v_s,t_s\right),\left(v_i, f\left(i\right)\right)\right\}$，$V_{\Pi_2}\left(t_s\right)=\left\{\left(v_s,t_s\right)\right\}$，标记源点$\left(v_s,t_s\right)$为 c，$m=0$，$L=\left\{v_s\right\}$；
4. While $Q\neq\phi$ do
5. select $\left(v_i, f\left(i\right)\right)$ from Q; $Q=Q-\left(v_i, f\left(i\right)\right)$;
6. If $v_i\neq v_d$ then
7. For each $e_j\in FS\left(v_i\right)$ do
8. $k_j=k_j+1$;
9. If $k_j=\left|T\left(e_j\right)\right|$ then
10. $t=\max\left\{f\left(x\right)\middle|v_x\in T\left(e_j\right)\right\}$;
11. For each $v_y\in H\left(e_j\right)$ and $v_y\notin L$ do
12. $f\left(y\right)=w_{e_j}\left(t\right)+t$;
13. EndFor
14. If $f\left(y\right)<T$ then
15. $p_v\left(v_y, f\left(y\right)\right)=e_j$; $L=L\cup\left\{v_y\right\}$;
16. 调用算法 3;
17. Else
18. If $Q\neq\phi$ then
19. return step 4;
20. Else
21. output no feasible solution;
22. EndIf
23. EndIf
24. EndIf
25. EndFor
26. Else
27. return $\mathscr{H}_{\Pi_1}\left(t_s\right)$，$\mathscr{H}_{\Pi_2}\left(t_s\right)$;
28. EndIf
29. EndWhile

4) 运行算法 3。当参数 m 为偶数时，遍历的节点$\left(v_y, f\left(y\right)\right)$属于路 $\mathscr{H}_{\Pi_1}\left(t_s\right)$，$m$ 为奇数时，节点

$\left(v_y, f\left(y\right)\right)$属于路$\mathscr{H}_{\Pi_2}\left(t_s\right)$。由于遍历的节点$\left(v_y, f\left(y\right)\right)$有两种情况，一种是属于$V_{\Pi}^{*}\left(t_s\right)$，一种是不属于$V_{\Pi}^{*}\left(t_s\right)$，需要分类讨论。

a) 当节点$\left(v_y, f\left(y\right)\right)$不属于$V_{\Pi}^{*}\left(t_s\right)$时，将节点放入$Q$中，然后返回算法2继续查询下一条超弧。

b) 当节点$\left(v_y, f\left(y\right)\right)$属于$V_{\Pi}^{*}\left(t_s\right)$时(算法3第9行)，节点$\left(v_y, f\left(y\right)\right)$是一个交错点。设$V_{\Pi}^{*}\left(t_s\right):c\to\left(v_y, f\left(y\right)\right)$为$\mathscr{H}_{\Pi}^{*}\left(t_s\right)$路上除$c$点和$\left(v_y, f\left(y\right)\right)$之外的它们之间经过的节点对。从点$\left(v_y, f\left(y\right)\right)$沿$\mathscr{H}_{\Pi}^{*}\left(t_s\right)$路反向找$V_{\Pi}^{*}\left(t_s\right):c\to\left(v_y, f\left(y\right)\right)$的节点时间对$\left\{\left(v_i, f\left(i\right)\right)\middle| v_i\neq v_s\neq v_y\right\}$，然后需要从节点$\left(v_i, f\left(i\right)\right)$继续往后查询，所以令集合$Q=\phi$，然后将$\left\{\left(v_i, f\left(i\right)\right)\right\}$放入集合$Q$中，并且放入集合$L$中，避免找路时头节点重复探查，即算法第11~12行。

接下来，当参数m为偶数时，将交错点之间的$V_{\Pi}^{*}\left(t_s\right)$加入到路径$\mathscr{H}_{\Pi_2}\left(t_s\right)$中的节点集$V_{\Pi_2}\left(t_s\right)$，即将$\mathscr{H}_{\Pi}^{*}\left(t_s\right)$路上$c\to\left(v_y, f\left(y\right)\right)$之间的路归于$\mathscr{H}_{\Pi_2}\left(t_s\right)$，$V_{\Pi_2}\left(t_s\right)=V_{\Pi_2}\left(t_s\right)\cup\left(V_{\Pi}^{*}\left(t_s\right):c\to\left(v_y, f\left(y\right)\right)\right)$，$m$为奇数时，将交错点之间的$V_{\Pi}^{*}\left(t_s\right)$加入到路径$\mathscr{H}_{\Pi_1}\left(t_s\right)$中，即将$\mathscr{H}_{\Pi}^{*}\left(t_s\right)$路上$c\to\left(v_y, f\left(y\right)\right)$之间的路归于$\mathscr{H}_{\Pi_1}\left(t_s\right)$，$V_{\Pi_1}\left(t_s\right)=V_{\Pi_1}\left(t_s\right)\cup\left(V_{\Pi}^{*}\left(t_s\right):c\to\left(v_y, f\left(y\right)\right)\right)$。最后由于节点$\left(v_y, f\left(y\right)\right)$是一个交错点，标记$\left(v_y, f\left(y\right)\right)$为$c$，用于之后查找不交路的连接，并更新参数$m$。接着返回算法2继续向后查询超路。

在算法过程中，当探查的节点为已知路$\mathscr{H}_{\Pi}^{*}\left(t_s\right)$的节点时间对时，我们需要沿着$\mathscr{H}_{\Pi}^{*}\left(t_s\right)$反向寻找交错点之间的节点，然后根据寻找的节点再出发寻求下一条不属于$\mathscr{H}_{\Pi}^{*}\left(t_s\right)$的超弧。算法的具体思路是在探查节点的过程中记录交错点数m，然后根据参数m的奇偶性对交错进行分类，分别对应$\mathscr{H}_{\Pi_1}\left(t_s\right)$，$\mathscr{H}_{\Pi_2}\left(t_s\right)$。

算法3：根据交错点找路

1. If m 为偶数 then $\left(v_y, f\left(y\right)\right)\in V_{\Pi_1}\left(t_s\right)$;
2. Else $\left(v_y, f\left(y\right)\right)\in V_{\Pi_2}\left(t_s\right)$;
3. EndIf
4. If $\left(v_y, f\left(y\right)\right)\notin V_{\Pi}^{*}\left(t_s\right)$ then
5. If $\left(v_y, f\left(y\right)\right)\notin Q$ then
6. $Q=Q\cup\left(v_y, f\left(y\right)\right)$;
7. return 算法2;
8. EndIf
9. Else
10. 从点$\left(v_y, f\left(y\right)\right)$沿$\mathscr{H}_{\Pi}^{*}\left(t_s\right)$路反向找$V_{\Pi}^{*}\left(t_s\right):c\to\left(v_y, f\left(y\right)\right)$的节点时间对$\left\{\left(v_i, f\left(i\right)\right)\middle| v_i\neq v_s\neq v_y\right\}$;
11. $Q\neq\phi$;
12. $Q=Q\cup\left\{\left(v_i, f\left(i\right)\right)\right\}$; $L=L\cup\left\{v_i\right\}$;
13. If m 为偶数 then
14. $V_{\Pi_2}\left(t_s\right)=V_{\Pi_2}\left(t_s\right)\cup\left(V_{\Pi}^{*}\left(t_s\right):c\to\left(v_y, f\left(y\right)\right)\right)$;
15. Else
16. $V_{\Pi_1}\left(t_s\right)=V_{\Pi_1}\left(t_s\right)\cup\left(V_{\Pi}^{*}\left(t_s\right):c\to\left(v_y, f\left(y\right)\right)\right)$;
17. EndIf
18. 标记$\left(v_y, f\left(y\right)\right)$为$c$;
19. $m=m+1$;
20. return 算法2;
21. EndIf

当算法 2 查询到的节点是目标节点 v_d 时，算法结束。根据算法过程得到两条动态限制不交 B-路 $\mathscr{H}_{\Pi_1}(t_s)$，$\mathscr{H}_{\Pi_2}(t_s)$。否则，算法找不到两条动态限制不交的 B-路，输出无结果。接下来我们通过一个例子来更直观的理解算法。

例 3 假设动态超图网络 $\mathscr{H}=(V(\mathscr{H}),E(\mathscr{H}),W,T)$。每条弧都有一个延迟函数 $w_e(t)$。节点 v_s 和 v_d 是源点和目标节点，出发时间 $t_s=0$，$L=\{v_s\}$，时间范围 $T=[0,50]$。假设通过算法 1 求得一条动态限制 B-路 $\mathscr{H}_\Pi^*(t_s)=(\{(v_s,0),(v_1,2),(v_2,1),(v_3,4),(v_5,5),(v_6,5),(v_d,7)\},\{e_1,e_2,e_3,e_5,e_8\})$。接下来我们可以根据以下步骤获得两条动态限制不交 B-路 $\mathscr{H}_{\Pi_1}(t_s)$，$\mathscr{H}_{\Pi_2}(t_s)$。见图 4。

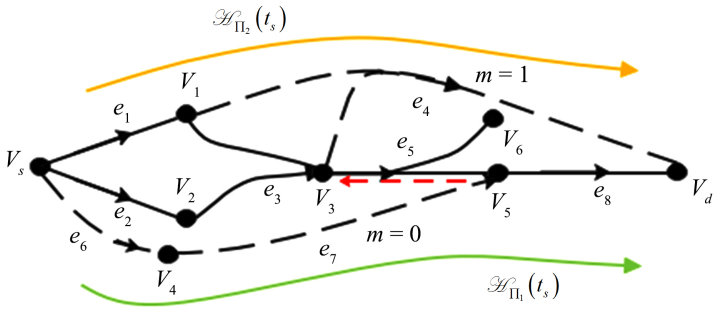

Figure 4. Dynamic hypergraph network $\mathscr{H}$
图 4. 动态超图网络 $\mathscr{H}$

算法过程：根据算法 2，初始状态 $Q=\{(v_4,f(4))|(v_s,v_4)\in E(\mathscr{H}),(v_4,f(4))\notin V_\Pi^*(t_s)\}$，其中 $f(4)=t_s+w_{s4}(t_s)=3$，路 $\mathscr{H}_{\Pi_1}(t_s)$ 的节点集 $V_{\Pi_1}(t_s)=\{(v_s,0),(v_4,3)\}$，路 $\mathscr{H}_{\Pi_2}(t_s)$ 的节点集 $V_{\Pi_2}(t_s)=\{(v_s,0)\}$，标记源点 $(v_s,0)$ 为 c，交错数 $m=0$。

算法 2 第 4 步，候选点集 $Q\neq\phi$，在集合 Q 中选择 $(v_4,f(4))$，并删除 $(v_4,f(4))$。选择 $e_7\in FS(v_4)$，$k_7=k_7+1=1$，满足 $k_7=|T(e_7)|$，故弧 e_7 的出发时间 $t=\max\{f(x)|v_x\in T(e_7)\}=3$。算法 2 第 11 行中，$v_5\in H(e_7)$ 并且 $v_5\notin L$，所以 $f(5)=w_{e_7}(3)+3=5$，$f(5)<50$，$p_v(v_5,5)=e_7$，$L=L\cup\{v_5\}$，进行算法 3。

由于此时 $m=0$ 为偶数，故将 $(v_5,5)$ 放入 $\mathscr{H}_{\Pi_1}(t_s)$ 中。因为 $(v_5,5)\in V_\Pi^*(t_s)$，故进行算法 3 第 9 步。从点 $(v_5,5)$ 沿 $\mathscr{H}_\Pi^*(t_s)$ 路反向找 $V_\Pi^*(t_s):(v_s,0)\to(v_5,5)$ 的节点时间对 $\{(v_i,f(i))|v_i\neq v_s\neq v_5\}$，由图 4 可知，节点时间对 $(v_1,2),(v_2,1),(v_3,4)$ 满足情况，$Q=\{(v_1,2),(v_2,1),(v_3,4)\}$，$L=L\cup\{v_1,v_2,v_3\}$。将 $V_\Pi^*(t_s):(v_s,0)\to(v_5,5)$ 之间的节点 $(v_1,2),(v_2,1),(v_3,4)$ 放入路 $\mathscr{H}_{\Pi_2}(t_s)$ 的节点集中，$V_{\Pi_2}(t_s)$ 标记 $(v_5,5)$ 为 c，令 $m=1$。

返回算法 2 继续查找下一个 Q 中的节点。最后，我们找到了两条限制不交的 B-路 $\mathscr{H}_{\Pi_1}(t_s)$，$\mathscr{H}_{\Pi_2}(t_s)$，其节点集为 $V_{\Pi_1}(t_s)=\{(v_s,0),(v_4,3),(v_5,5),(v_d,7)\}$，$V_{\Pi_2}(t_s)=\{(v_s,0),(v_1,2),(v_2,1),(v_3,4),(v_d,7)\}$。超弧集为 $E_{\Pi_1}(t_s)=\{e_6,e_7,e_8\}$，$E_{\Pi_2}(t_s)=\{e_1,e_2,e_3,e_4\}$。

3.2.3. 算法正确性和复杂性证明

这一节，我们给出定理 2 说明算法的正确性，并证明算法复杂性。

定理 2 算法 2 输出两条动态限制不交 B-路 $\mathscr{H}_{\Pi_1}(t_s)$，$\mathscr{H}_{\Pi_2}(t_s)$ 当且仅当超图 $\mathscr{H}$ 中存在问题 2 的解。

证明：算法的目的是为了找两条动态限制不交 B-路。当算法 2 输出 $\mathscr{H}_{\Pi_1}(t_s)$，$\mathscr{H}_{\Pi_2}(t_s)$，说明超图 $\mathscr{H}$ 在时间范围 $[0,T]$ 内可以找到两条动态限制不交 B-路。首先，在 3.2.1 部分的准备工作中，我们更新路 $\mathscr{H}_\Pi^*(t_s)$ 上所有超弧 e_j 的延迟函数 $w_e(t)$，对路 $\mathscr{H}_\Pi^*(t_s)$ 上超弧经过的所有时间段的函数值赋值为 ∞。这一操作是为了在找路的过程中，避开路 $\mathscr{H}_\Pi^*(t_s)$ 上所有弧的经过时间段，保证了两条路是动态边不交的。

并且在算法 2 第 14 行，节点的到达时间满足条件 $f(y)<T$。对于节点时间对相交，算法提出交错的概念，在探查节点的过程中记录交错点数 m，然后根据参数 m 的奇偶性对探查的超弧进行分类，分别对应 $\mathscr{H}_{\Pi_1}(t_s)$，$\mathscr{H}_{\Pi_2}(t_s)$。对于每一个探查过的节点，都将其放入集合 L 中，保证了所找的路不存在圈。故算法所找出来的是两条动态限制不交的 B-路。

另外，由于在调用算法 3 时，算法 10~12 行考虑了交错点之间所有以 $\mathscr{H}_{\Pi}^{*}(t_s)$ 上的点为尾节点出去的超弧，即算法遍历了所有满足条件的超弧。所以，当算法 2 输出不存在时，说明超图 $\mathscr{H}$ 在时间范围 $[0,T]$ 内没有满足条件的动态限制不交 B-路。否则，如果存在，在运行算法 3 的第 10~12 行更新集合 Q 后，算法 2 进行第 4~16 行时一定会找到满足条件的路径。

定理 3 算法 2 的复杂度为 $O\left(n\cdot\alpha(T)size(\mathscr{H})\right)$。

证明： 候选点集 Q 中的元素最多有 $n\cdot\alpha(T)$ 个节点时间对，对于每一个探查过的节点，都将其放入集合 L 中，所以对于超弧头节点和尾节点的探查总共需要 $size(\mathscr{H})$ 的时间。算法 3 需要 $O(1)$ 的运行时间。故算法 2 的复杂度为 $O\left(n\cdot\alpha(T)size(\mathscr{H})\right)$。

3.2.4. 算法的推广

本文考虑的是连续时间动态超图网络中的动态限制不交 B-路，相应的我们可以获得动态不交 F-路。由于 F-弧满足的条件是 $|T(e)|=1$，所以我们在计算路的延迟时，考虑每条超弧的尾节点的到达时间为超弧的出发时间，显然到达头节点的时间都是相等的，并且在探查到交错点时，同样沿 $\mathscr{H}_{\Pi}^{*}(t_s)$ 路反向找到超弧对应的尾节点，然后从尾节点继续向后探查。可以看出，求动态限制不交 F-路的算法与求动态限制不交 B-路的算法类似，在这里，我们就不再给出算法。

4. 结语

结合现实中的实际问题，本文提出了一个新的网络环境，连续时间动态超图网络。动态超图网络与传统的网络模型相比应用范围更广，也更有实际意义。本文在连续时间动态超图网络中，通过对超图特殊性的分析，设计了求解动态限制不交 B-路问题的算法，证明算法是在多项式时间 $O\left(n\cdot\alpha(T)size(\mathscr{H})\right)$ 内可解决的。在动态超图网络中还可以进一步研究在其他领域的应用，求解不同方面的问题。

基金项目

山西省自然科学基金(202103021224058)。

参考文献

[1] Marcotte, P. and Nguyen, S. (1998) Hyperpath Formulations of Traffic Assignment Problems. In: Marcotte, P. and Nguyen, S., Eds., *Equilibrium and Advanced Transportation Modelling*, Springer, Boston, 175-200. https://doi.org/10.1007/978-1-4615-5757-9_9

[2] Nguyen, S., Pallottino, S. and Gendreau, M. (1998) Implicit Enumeration of Hyperpaths in a Logit Model for Transit Networks. *Transportation Science*, **32**, 54-64. https://doi.org/10.1287/trsc.32.1.54

[3] Pretolani, D. (2000) A Directed Hypergraph Model for Random Time Dependent Shortest Paths. *European Journal of Operational Research*, **123**, 315-324. https://doi.org/10.1016/S0377-2217(99)00259-3

[4] Boley, H. (1977) Directed Recursive Labelnode Hypergraphs: A New Representation-Language. *Artificial Intelligence*, **9**, 49-85. https://doi.org/10.1016/0004-3702(77)90014-5

[5] Torres, A. and Araoz, J. (1988) Combinatorial Models for Searching in Knowledge Bases. *Acta Científica Venezolana*, **39**, 387-394.

[6] Ausiello, G., D'Atri, A. and Saccà, D. (1983) Graph Algorithms for Functional Dependency Manipulation. *Journal of the ACM* (*JACM*), **30**, 752-766. https://doi.org/10.1145/2157.322404

[7] Carraresi, P., Gallo, G. and Rago, G. (1993) A Hypergraph Model for Constraint Logic Programming and Applications

to Bus Drivers' Scheduling. *Annals of Mathematics and Artificial Intelligence*, **8**, 247-270. https://doi.org/10.1007/BF01530792

[8] Gallo, G. and Scutella, M.G. (1998) Directed Hypergraphs as a Modelling Paradigm. *Rivista di Matematica per le Scienze Economiche e Sociali*, **21**, 97-123. https://doi.org/10.1007/BF02735318

[9] Italiano, G.F. and Nanni, U. (1989) Online Maintenance of Minimal Directed Hypergraphs. 3*rd Italian Conference on Theoretical Computer Science*, Mantova, 2-4 November 1989, 335-349.

[10] Gallo, G., Longo, G., Palottino, S. and Nguyen, S. (1993) Directed Hypergraphs and Applications. *Discrete Applied Mathematics*, **42**, 177-201. https://doi.org/10.1016/0166-218X(93)90045-P

[11] Suurballe, J.W. (1974) Disjoint Paths in a Network. *Networks*, **4**, 125-145. https://doi.org/10.1002/net.3230040204

[12] Suurballe, J.W. and Tarjan, R.E. (1984) A Quick Method for Finding Shortest Pairs of Disjoint Paths. *Networks*, **14**, 325-336. https://doi.org/10.1002/net.3230140209

[13] Yang, B., Zheng, S.Q. and Katukam, S. (2003) Finding Two Disjoint Paths in a Network with Min-Min Objective Function. Parallel and Distributed Computing and Systems Vol.1, Department of Computer Science, University of Texas at Dallas, Richardson.

[14] Fleischer, R., Ge, Q., Li, J. and Zhu, H. (2007) Efficient Algorithms for *k*-Disjoint Paths Problems on DAGs. *International Conference on Algorithmic Applications in Management*, Portland, 6-8 June 2007, 134-143. https://doi.org/10.1007/978-3-540-72870-2_13

[15] Fortune, S., Hopcroft, J. and Wyllie, J. (1980) The Directed Subgraph Homeomorphism Problem. *Theoretical Computer Science*, **10**, 111-121. https://doi.org/10.1016/0304-3975(80)90009-2

[16] Parpalea, M. and Ciurea, E. (2011) The Quickest Maximum Dynamic Flow of Minimum Cost. *International Journal of Applied Mathematics and Informatics*, **5**, 266-274.

[17] Cai, X., Sha, D. and Wong, C.K. (2001) Time-Varying Minimum Cost Flow Problems. *European Journal of Operational Research*, **131**, 352-374. https://doi.org/10.1016/S0377-2217(00)00059-X

[18] Ding, B., Yu, J.X. and Qin, L. (2008) Finding Time-Dependent Shortest Paths over Large Graphs. *Proceedings of the 11th International Conference on Extending Database Technology*: *Advances in Database Technology*, Nantes, 25-29 March 2008, 205-216. https://doi.org/10.1145/1353343.1353371

[19] Wang, Y., Li, G. and Tang, N. (2019) Querying Shortest Paths on Time Dependent Road Networks. *Proceedings of the VLDB Endowment*, **12**, 1249-1261. https://doi.org/10.14778/3342263.3342265.