文章编号: 1001-0920(2009)02-0274-05

动态搜索算法求解时间依赖型旅行商问题研究

李妍峰[1,2], 李　军[1], 赵　达[1]
(1. 西南交通大学 经济管理学院, 成都 610031; 2. 四川师范大学 商学院, 成都 610072)

摘　要: 时间依赖型旅行商问题(TDTSP)是旅行商问题(TSP)的延伸. 在该问题中,任意两节点间的旅行时间(成本)不仅取决于节点间的距离,还依赖于一天中具体时段或节点在哈密顿圈中所处的具体位置. 对基于节点所处哈密顿圈中具体位置的 TDTSP 问题建立相应的数学模型,并提出求解该问题的动态搜索算法. 通过实验仿真,验证了动态搜索算法优于目前在邻域搜索领域求解该问题最有效的动态规划启发式算法.

关键词: 时间依赖型旅行商问题; 哈密顿圈; 动态搜索算法; 动态规划启发式
中图分类号: F253.4　　**文献标识码**: A

Dynasearch algorithms for solving time dependent traveling salesman problem

LI Yan-feng[1,2], *LI Jun*[1], *ZHAO Da*[1]
(1. School of Economics and Management, Southwest Jiaotong University, Chengdu 610031, China; 2. Business School, Sichuan Normal University, Chengdu 610072, China. Correspondent: LI Yan-feng, E-mail: yanwaa@126.com)

Abstract: Time dependent traveling salesman problem (TDTSP) is an extension of the traveling salesman problem (TSP), in which the travel time or cost between two nodes depends on not only the distance between the nodes, but also the time of day or the node position in the Hamilton cycle. In this paper, we formulate the model for the TDTSP based on the node position in the Hamilton cycle, and develop the novel dynasearch algorithms to solve it. Simulation shows that the dynasearch algorithms are superior to the most effective dynamic programming heuristics in local search area.

Key words: Time dependent traveling salesman problem; Hamilton cycle; Dynasearch algorithms; Dynamic programming heuristics

1 引　　言

旅行商问题(TSP)是运筹学和组合优化领域最经典的问题之一,长期以来一直是该领域的研究热点. 一般 TSP 问题中任意两节点间的旅行时间已知且恒定. 本文中研究的时间依赖型旅行商问题(TDTSP)是 TSP 问题的一类延伸. 在实际中,由于受交通事故、天气变化、上下班高峰期和道路特征的影响,任意两节点间的旅行时间依赖于一天中具体时段或节点在哈密顿圈中所处的具体位置,因此研究该问题更具有实际意义.

目前,对 TDTSP 问题的处理方法大致可分为两类:1) 旅行商在两节点间的旅行时间依赖于一天中的具体时段,该类问题广泛存在于城市交通网络中,车辆的行驶时间会受到交通管理、交通流量、交通事故、上下班高峰期等因素的影响而变化,在不同时段内任意两节点间的旅行时间不同,会出现跨时段现象[1-10];2) 旅行商在路段 (i, j) 的旅行时间依赖于节点 i 在哈密顿圈中所处的具体位置,该问题相当于任意两节点间的旅行时间为一时段. 实际中也存在很多应用,如旅行修理工问题中需要考虑服务对象的紧急性,生产调度问题中需要考虑机器的疲劳效应或生产条件变更等因素,城市交通问题中需要考虑路段特征等. 该类问题在旅行过程中对任意路段都不会出现跨时段现象[11-15],但需要数据非常庞大. 假设有 N 个节点,则需要生成 N^3 个数据,在计算空间和时间方面受到很大的限制.

收稿日期: 2007-12-29; **修回日期**: 2008-04-10.
基金项目: 国家社科基金项目(07BJY038); 教育部新世纪优秀人才支持计划项目(NCET-04-0886).
作者简介: 李妍峰(1980—),女,四川乐山人,讲师,博士生,从事运筹学、物流优化的研究; 李军(1967—),女,成都人,教授,博士生导师,从事运筹学、物流优化等研究.

在算法方面,由于 TDTSP 为 NP 难题,Fox[11] 指出分支定界算法在 12 min 内甚至不能求解 10 个工件的问题,其应用范围非常有限,因此大部分研究者考虑如何构造启发式算法进行求解. Malandraki 等[5] 修正了动态规划精确算法,通过限制每阶段的状态数量 H,提出了动态规划启发式算法,并求解了 55 个节点的 TDTSP 问题. 该算法实际上是动态规划精确算法和最近邻算法的综合,是目前在邻域搜索领域求解 TDTSP 问题最有效的启发式算法. 但利用该算法求解的问题规模也很小,不能为实际问题提供实质性的决策支持.

本文对依赖于节点所处哈密顿圈中具体位置的 TDTSP 问题建立了数学模型,并构造了适合求解该问题的动态搜索算法. 最后进行了实验仿真,与动态规划启发式算法进行了性能比较.

2　TDTSP 问题模型

本文以从最小化节点 1 出发,经过剩余所有节点,并回到节点 1 的时刻作为问题目标. 假设在该类 TDTSP 中共有 N 个节点,为了构成哈密顿圈,令节点 $N+1$ 为节点 1. t_i 为旅行商到达节点 i 的时刻,因不考虑在节点处停留,t_i 也为旅行商从节点 i 出发的时刻,c_{ijp} 为旅行商从节点 i 出发到节点 j 的旅行时间,此时节点 i 在哈密顿圈中所处第 p 个位置.

模型如下:

$$\text{Min } t_{N+1}. \tag{1}$$

s.t.

$$\sum_{i=1,i\neq j}^{N}\sum_{p}^{N} x_{ijp} = 1,\ j = 2,\cdots,N+1; \tag{2}$$

$$\sum_{j=2,j\neq i}^{N+1}\sum_{p}^{N} x_{ijp} = 1,\ i = 1,\cdots,N; \tag{3}$$

$$t_j - t_i - L x_{ijp} \geq c_{ijp} - L,\ i = 1,\cdots,N,\ j = 2,\cdots,N+1,\ i \neq j,\ p = 1,\cdots,N; \tag{4}$$

$$\sum_{i}^{N}\sum_{j}^{N+1}\sum_{p}^{N} x_{ijp} \leq |S| - 1,\ \forall S \subseteq \{1,\cdots,N\} \text{ 且 } |S| \geq 2; \tag{5}$$

$$\sum_{i}^{N} x_{ijp} = \sum_{k}^{N+1} x_{jk,p+1},\ j = 1,\cdots,N,\ p = 1,\cdots,N-1; \tag{6}$$

$$t_i \geq 0,\ i = 1,\cdots,N+1. \tag{7}$$

式(1) 为最小化到达节点 $N+1$ 的时刻,即访问完所有节点并回到节点 1 的时刻. 式(2) 保证无论节点 i 所处任何位置,节点 j 紧前只有一个节点,其中决策变量

$$x_{ijp} = \begin{cases} 1, & \text{若旅行商在第 } p \text{ 个位置访问节点 } i, \\ & \text{并从节点 } i \text{ 出发到节点 } j; \\ 0, & \text{否则.} \end{cases}$$

式(3) 保证无论节点 i 所处任何位置,节点 i 紧后只有一个节点. 式(4) 计算从节点 i 出发,到达节点 j 的时刻,其中 L 为一任意大数. 当 $x_{ijp} = 1$ 时,该式取等号. 式(5) 为 TDTSP 问题的避免子环约束. 式(6) 保证路径连续. 式(7) 保证到达任一节点的时刻非负.

3　动态搜索算法

Congram[16] 提出求解一般 TSP 的 dynasearch 策略,该策略包括 3 种算法:ds-2-opt,ds-2.5-opt 和 ds-3-opt,分别采用动态规划算法搜索 TSP 的 2-opt[17],or-opt[18] 和 3-opt[17] 邻域. 该策略在多项式时间范围内搜索到指数形式大小的邻域,但 Congram[16] 只求解了对称的 TSP 问题,即旅行商在任意路段 (i,j) 和 (j,i) 的旅行时间相同,且没考虑旅行时间在节点所处不同位置时的变化. 本文利用 dynasearch 策略的思想,提出求解基于具体位置的 TDTSP 问题的动态搜索算法. 假设初始路径为 $(\sigma(1),\sigma(2),\cdots,\sigma(N+1))$,其中 $\sigma(j)$ 表示初始路径第 j 个位置的节点. 为构成哈密顿圈,令 $\sigma(N+1) = \sigma(1)$.

(1) Ds-2-opt

2-opt 移动以任意一条路径开始,断开其任意两边,以另外一种方式连接断开部分,构成一条新的路径,若旅行商在该新路径旅行时间比原路径的时间短,则替换原路径[17]. ds-2-opt 算法对当前路径执行一系列独立 2-opt 移动. 给定路径 $(1,2,\cdots,i,i+1,\cdots,j,j+1,\cdots,k,k+1,\cdots,s,s+1,\cdots,N+1)$,分别用路段 (i,j) 和路段 $(i+1,j+1)$ 替换路段 $(i,i+1)$ 和路段 $(j,j+1)$,同时分别用路段 (k,s) 和路段 $(k+1,s+1)$ 替换路段 $(k,k+1)$ 和路段 $(s,s+1)$,其中 $1 \leq i < j \leq N+1$,$1 \leq k < s \leq N+1$,当 $j < k$ 时称为独立移动.

假设旅行商从节点 $\sigma(1)$ 出发的时刻为 t_1,$t_{\sigma(j)}$ 为旅行商从节点 $\sigma(1)$ 出发,到达节点 $\sigma(j)$ 的最早时刻,也为旅行商从节点 $\sigma(j)$ 出发的最早时刻 $(j = 1,2,\cdots,N+1)$. 每次迭代过程中在节点 $\sigma(j)$ 后加入 1 个节点,则 $t_{\sigma(j+1)}$ 对应到达节点 $\sigma(j+1)$ 的最早时刻. 当前路径有 2 种方式加入节点 $\sigma(j+1)$:①节点 $\sigma(j)$ 后加入节点 $\sigma(j+1)$,并保持原路径顺序不变(见图 1);②加入节点 $\sigma(j+1)$ 后,对路段 $(\sigma(i),\sigma(j))$ 和路段 $(\sigma(i+1),\sigma(j+1))$ 进行 2-opt 移动(见图 2,$i = 1,\cdots,j-2$.

初始条件如下:

$$t_{\sigma(1)} = t_1, \tag{8}$$

$$t_{\sigma(2)} = t_{\sigma(1)} + c_{\sigma(1),\sigma(2),1}, \tag{9}$$

$$t_{\sigma(3)} = t_{\sigma(2)} + c_{\sigma(2),\sigma(3),2}. \tag{10}$$

对 $j = 3,\cdots,N$,递归公式

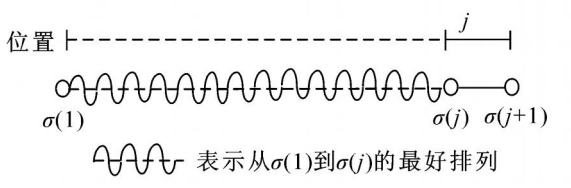

图 1 ds-2-opt 方式 ①邻域变换

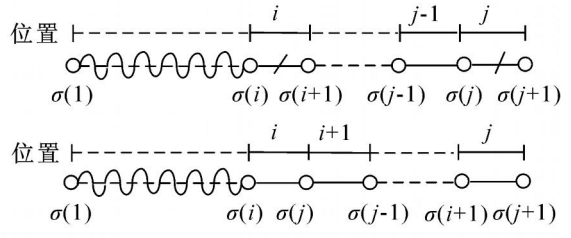

图 2 ds-2-opt 方式 ②邻域变换

$$t_{\sigma(j+1)} = \min\begin{cases} t_{\sigma(j)} + c_{\sigma(j),\sigma(j+1),j}, \\ \min\limits_{1\le i\le j-2}\{t_{\sigma(i)} + c_{\sigma(i),\sigma(j),i} + \\ \qquad D_{\sigma(i+1),\sigma(j)} + c_{\sigma(i+1),\sigma(j+1),j}\}. \end{cases} \tag{11}$$

其中 $D_{\sigma(i+1),\sigma(j)} = \sum_{k=j}^{i+2} c_{\sigma(k),\sigma(k-1),j+i+1-k}$. 第 $j+1$ 阶段最优值为方式 ①和 ②中到达节点 $\sigma(j+1)$ 的最早时刻.

(2) Ds-2.5-opt

ds-2.5-opt 邻域将 2-opt 邻域和 or-opt 邻域相结合. 在 ds-2-opt 的基础上,采用前向插入和后向插入策略,动态规划搜索过程同 ds-2-opt. 当前路径有 4 种方式加入节点 $\sigma(j+1)$:方式 ①和 ②同 ds-2-opt;方式 ③在节点 $\sigma(j)$ 后加入节点 $\sigma(j+1)$,将节点 $\sigma(j)$ 插入到节点 $\sigma(i)$ 和节点 $\sigma(i+1)$ 之间(见图 3),$i = 1, \cdots, j-2$;方式 ④在节点 $\sigma(j)$ 后加入节点 $\sigma(j+1)$,将节点 $\sigma(i+1)$ 插入到节点 $\sigma(j)$ 和节点 $\sigma(j+1)$ 之间(见图 4),$i = 1, \cdots, j-2$.

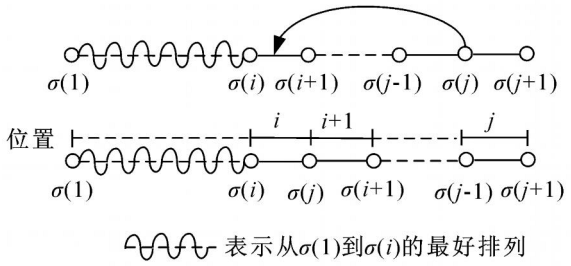

图 3 ds-2.5-opt 方式 ③邻域变换

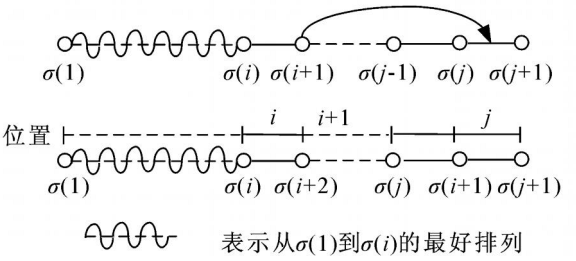

图 4 ds-2.5-opt 方式 ④邻域变换

初始条件同 ds-2-opt.

对 $j = 3, \cdots, N$,递归公式

$$t_{\sigma(j+1)} = \min\begin{cases} t_{\sigma(j)} + c_{\sigma(j),\sigma(j+1),j}, \\ \min\limits_{1\le i\le j-2}\{t_{\sigma(i)} + c_{\sigma(i),\sigma(j),i} + \\ \qquad D_{\sigma(i+1),\sigma(j)} + c_{\sigma(i+1),\sigma(j+1),j}\}, \\ \min\limits_{1\le i\le j-2}\{t_{\sigma(i)} + c_{\sigma(i),\sigma(j),i} + c_{\sigma(j),\sigma(i+1),i+1} + \\ \qquad \sum_{k=i+2}^{j-2} c_{\sigma(k),\sigma(k+1),k+1} + c_{\sigma(j-1),\sigma(j+1),j}\}, \\ \min\limits_{1\le i\le j-2}\{t_{\sigma(i)} + c_{\sigma(i),\sigma(i+2),i} + \sum_{k=i+2}^{j-1} c_{\sigma(k),\sigma(k+1),k-1} + \\ \qquad c_{\sigma(j),\sigma(i+1),j-1} + c_{\sigma(i+1),\sigma(j+1),j}\}. \end{cases} \tag{12}$$

递归公式中前两个表达式与 ds-2-opt 相同,后两个分别对应 or-opt 中的前向插入和后向插入. 第 $j+1$ 阶段的最优值为方式 ①～ ④到达节点 $\sigma(j+1)$ 的最早时刻.

(3) Ds-3-opt

3-opt 移动以任意一条路径开始,断开其任意 3 条边,以其他方式重新连接断开部分,构成一条新的路径. 若旅行商在该新路径旅行时间比原路径的时间短,则替换原路径[17]. 重新连接路径有 4 种不同的方式,因此在执行 ds-3-opt 过程中,加入节点 $\sigma(j+1)$ 有 5 种不同的方式:①在节点 $\sigma(j)$ 后加入节点 $\sigma(j+1)$,并保持原路径不变;②～ ⑤在节点 $\sigma(j)$ 后加入节点 $\sigma(j+1)$,删除 3 条路段 $(\sigma(h),\sigma(h+1))$,$(\sigma(i),\sigma(i+1))$ 和 $(\sigma(j),\sigma(j+1))$,执行 3-opt 移动,其中 $1 \le h \le i-2 \le j-4$,并分别采用不同的 4 种方式连接(见图 5).

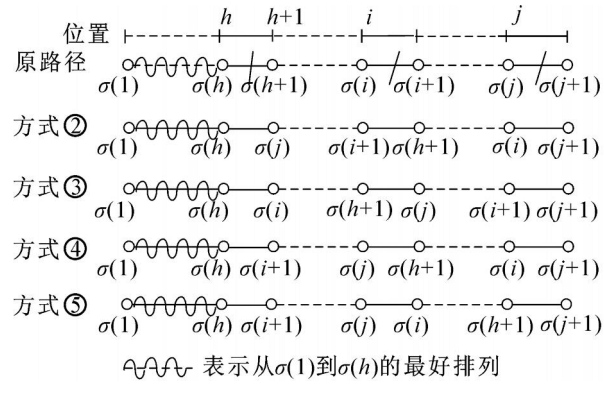

图 5 ds-3-opt 方式 ②～ ⑤邻域变换

初始条件如下:前 3 个节点初始化同 ds-2-opt 的初始化. $j = 3$,在节点 $\sigma(3)$ 后加入节点 $\sigma(4)$,有

$$t_{\sigma(4)} = t_{\sigma(3)} + c_{\sigma(3),\sigma(4),3}. \tag{13}$$

对 $j = 4, \cdots, N$,方式 ①的状态值可表示为 $t_{\sigma(j)}$

$+ c_{\sigma(j),\sigma(j+1),j}$. 对方式 ②,路径中共有 2 段虚线. 根据第 1 段虚线内弧段数目与对应节点所处位置数目相等,可得 $j-(i+1)=i-(h+1)$,经整理得 $h=2i-j$.

第 1 段虚线所对应的旅行时间可表示为

$$\sum_{k=j}^{i+2} c_{\sigma(k),\sigma(k-1),2i+1-k}. \tag{14}$$

同理,针对第 2 段虚线,根据第 1 段虚线内弧段数目与对应节点所处位置数目相等,可得 $i-(h+1)=j-(i+1)$,经整理,同样得到 $h=2i-j$.

第 2 段虚线所对应的旅行时间可表示为

$$\sum_{k=h+1}^{i-1} c_{\sigma(k),\sigma(k+1),i-h+k}. \tag{15}$$

通过以上分析,可得到 ds-3-opt 方式 ②～ ⑤对应虚线部分的旅行时间,如表 1 所示.

递归公式如下:

$$t_{\sigma(j+1)}=\min\begin{cases} t_{\sigma(j)}+c_{\sigma(j),\sigma(j+1),j}, \\ \min\limits_{1\le h\le i-2\le j-4}\{t_{\sigma(h)}+c_{\sigma(h),\sigma(j),h}+D_{11}+c_{\sigma(i+1),\sigma(h+1),i}+D_{12}+c_{\sigma(i),\sigma(j+1),j}\}, \\ \min\limits_{1\le h\le i-2\le j-4}\{t_{\sigma(h)}+c_{\sigma(h),\sigma(i),h}+D_{21}+c_{\sigma(h+1),\sigma(j),i}+D_{22}+c_{\sigma(i+1),\sigma(j+1),j}\}, \\ \min\limits_{1\le h\le i-2\le j-4}\{t_{\sigma(h)}+c_{\sigma(h),\sigma(i+1),h}+D_{31}+c_{\sigma(j),\sigma(h+1),i}+D_{32}+c_{\sigma(i),\sigma(j+1),j}\}, \\ \min\limits_{1\le h\le i-2\le j-4}\{t_{\sigma(h)}+c_{\sigma(h),\sigma(i+1),h}+D_{41}+c_{\sigma(j),\sigma(i),i}+D_{42}+c_{\sigma(h+1),\sigma(j+1),j}\}. \end{cases} \tag{16}$$

第 $j+1$ 阶段的最优值为方式 ①～ ⑤中到达节点 $\sigma(j)$ 的最早时刻.

当加入节点 $\sigma(N+1)$ 后,通过计算得到最优值 $t_{\sigma(N+1)}$,以上 3 种算法均通过动态规划回溯找到最优路径. 从 ds-2-opt 和 ds-2.5-opt 递归方程可以发现,对每一个状态,从其他状态转移到该状态的计算最大维数为 $N-2$,总共需要计算 $N+1$ 个状态,所以计算复杂度为 $(N+1)(N-2)$,表示为 $O(N^2)$;在 ds-3-opt 中,对每一个状态,从其他状态转移到该状态的计算最大维数为 $(N-4)(N-2)$,总共需要计算 N 个状态,计算复杂度为 $(N+1)(N-4)(N-2)$,表示为 $O(N^3)$.

表 1　ds-3-opt 方式 ②～ ⑤邻域变换第 1,2 段虚线旅行时间

变换方式	第 1 段虚线旅行时间	第 2 段虚线旅行时间
②	$D_{11}=\sum_{k=j}^{i+2} c_{\sigma(k),\sigma(k-1),2i+1-k}$	$D_{12}=\sum_{k=h+1}^{i-1} c_{\sigma(k),\sigma(k+1),i-h+k}$
③	$D_{21}=\sum_{k=i}^{h+2} c_{\sigma(k),\sigma(k-1),i+h+1-k}$	$D_{22}=\sum_{k=j}^{i+2} c_{\sigma(k),\sigma(k-1),i+j+1-k}$
④	$D_{31}=\sum_{k=i+1}^{j-1} c_{\sigma(k),\sigma(k+1),i-j+k}$	$D_{32}=\sum_{k=h+1}^{i-1} c_{\sigma(k),\sigma(k+1),i-h+k}$
⑤	$D_{41}=\sum_{k=i+1}^{j-1} c_{\sigma(k),\sigma(k+1),i-j+k}$	$D_{42}=\sum_{k=i}^{h+2} c_{\sigma(k),\sigma(k-1),2i+1-k}$

4　实验及算法性能比较

本文用 10 组不同规模的随机产生数据分别测试动态搜索算法和动态规划启发式算法[5]. 所有程序均用 C++ 编写,并在 Pentium Ⅳ主频 1.6 GHz 的计算机上进行实验仿真. 表 2 记录了各类算法的实验规模与实验结果(包括目标函数值和算法运行时间).

由表 2 得出如下结论:

1) 随着 TDTSP 问题节点数的增加,即问题规模的扩大,每一类算法的运行时间随之增加.

2) 就动态规划启发式算法而言,$H=1$(即每阶段只选一个最好的状态,相当于最近邻算法) 时目标函数值劣于 $H=2$(即每阶段选两个最好的状

表 2　动态搜索算法与动态规划启发式算法性能比较

数据规模(节点数)	动态搜索算法						动态规划启发式算法			
	ds-2-opt		ds-2.5-opt		ds-3-opt		$H=2$		$H=1$	
	目标函数值/h	运行时间/s	目标函数值/h	运行时间/s	目标函数值/h	运行时间/s	目标函数值/h	运行时间/s	目标函数值/h	运行时间/s
5	5.38	0.00	5.24	0.00	5.28	0.00	5.80	0.02	6.10	0.00
10	9.32	0.00	8.52	0.00	7.80	0.01	7.98	0.01	8.58	0.02
20	17.42	0.03	15.62	0.07	15.28	0.06	16.91	0.15	17.20	0.06
30	26.33	0.20	22.80	0.19	21.84	0.08	22.82	0.26	26.62	0.37
50	42.54	0.30	38.12	0.41	32.13	0.57	34.19	0.85	37.57	1.03
60	49.40	0.50	45.67	0.71	36.51	0.80	39.69	1.61	47.61	1.58
80	64.32	1.21	60.54	1.03	56.32	1.55	NA		63.42	2.39
100	82.35	1.44	78.91	1.82	69.43	2.07	NA		85.17	3.83
120	101.73	1.83	95.32	2.46	88.91	3.33	NA		NA	
150	133.94	2.25	120.87	2.99	103.23	4.06	NA		NA	
平均	53.27	0.776	49.16	0.968	43.67	1.253				

注:NA 表示不能运行计算.

态),但前者运行时间比后者少,且前者求解的最大规模大于后者.

3) 动态搜索算法与动态规划启发式算法相比,由于 ds-2-opt 的邻域空间范围较小,因此解的质量不如动态规划启发式算法. 随着动态搜索算法邻域空间的扩大,部分 ds-2.5-opt 解和绝大部分 ds-3-opt 解优于动态规划启发式算法解. 随着问题规模增大,导致动态规划启发式算法的状态空间扩大,超过了存储空间容量,从而使得该算法不能求解.

4) 在表 2 中最后一行,给出了动态搜索策略各算法的平均目标函数值和平均运行时间. ds-2.5-opt 的平均目标函数值比 ds-2-opt 的平均目标函数值少 7.71 %,ds-3-opt 的平均目标函数值比 ds-2-opt 的平均目标函数值少 18.02 %; 就运行时间而言,ds-2.5-opt 运行时间比 ds-2-opt 的运行时间多 19.83 %,由于 ds-3-opt 邻域空间的扩大,使得运行时间比前两种长.

5 结 论

本文对一类旅行时间依赖于节点在哈密顿圈所处位置的 TDTSP 问题建立了数学模型,并构造了适合求解该问题的动态搜索算法 ds-k-opt (k = 2, 2.5, 3). 通过实验仿真比较,在求解规模上动态搜索算法能求解更大规模的 TDTSP 问题,在求解质量上部分 ds-2.5-opt 解和绝大部分 ds-3-opt 解优于动态规划启发式算法解. 就动态搜索算法的 3 种类型而言,解随 k 值的不断增大而更优,但算法运行时间随 k 值的不断增大而更长. 该方法还可应用于时变车辆调度问题和时变机器调度问题.

参考文献(References)

[1] Malandraki C. Time dependent vehicle routing problem: Formulations, solution algorithms and computations experiments[D]. Evanston: Northwestern University, 1989.

[2] Ahn B H, Shin J Y. Vehicle-routing with time windows and time-varying congestion[J]. J of the Operational Research Society, 1991, 42(5): 393-400.

[3] Hill A V, Benton W C. Modeling intra-city time-dependent travel speeds for vehicle scheduling problems [J]. J of the Operational Research Society, 1992, 43(4): 343-351.

[4] Malandraki C, Daskin M S. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms[J]. Transportation Science, 1992, 26(3): 185-200.

[5] Malandraki C, Dial R B. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem [J]. European J of Operational Research, 1996, 90(1): 45-55.

[6] Horn M E T. Efficient modeling of travel in networks with time-varying link speeds[J]. Networks, 2000, 36(2): 80-90.

[7] Ichoua S, Gendreau M, Potvin J Y. Vehicle dispatching with time-dependent travel times[J]. European J of Operational Research, 2003, 144(2): 379-396.

[8] Fleischmann B, Gietz M, Gnutzmann S. Time-varying travel times in vehicle routing[J]. Transportation Science, 2004, 38(2): 160-173.

[9] Haghani A, Jung S. A dynamic vehicle routing problem with time-dependent travel times[J]. Computer & Operations Research, 2005, 32(11): 2959-2986.

[10] Li F Y. Modeling and solving variants of the vehicle routing problem: Algorithms, test problems, and computational results[D]. Baltimore: University of Maryland, 2005.

[11] Fox K R, Gavish B, Graves S C. A n-constraint formulation of the (time-dependent) traveling salesman problem[J]. Operations Research, 1980, 28(4): 1018-1021.

[12] Picard J C, Queryranne M. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling[J]. Operations Research, 1978, 26(1): 86-110.

[13] Lucena A. Time-dependent traveling salesman problem — The deliveryman case[J]. Networks, 1990, 20(66): 753-763.

[14] Bianco L, Mingozzi A, Ricciardelli S. The traveling salesman problem with cumulative cost[J]. Networks, 1993, 23(2): 81-91.

[15] Wiel R J V, Sahinidis N V. Heuristic bounds and test problem generation for the time-dependent traveling salesman problem[J]. Transportation Science, 1995, 29(22): 167-183.

[16] Congram R K. Polynominally searchable exponential neighbourhoods for sequencing problems in combinatorial optimization [D]. Southampton: University of Southampton, 2000.

[17] Lin S, Kernighan B. An effective heuristic algorithm for the traveling salesman problem[J]. Operations Research, 1973, 21(2): 498-516.

[18] Or I. Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking[D]. Evanston: Northwestern University, 1976.