一类求解带时间窗的团队定向问题的改进蚁群算法

柯良军 章 鹤 尚 可 冯祖仁
(西安交通大学机械制造与系统工程国家重点实验室 西安 710049)

摘 要 带时间窗的团队定向问题是一类重要的物流配送路径优化问题,其优化目标是制定最优可行车辆路线,在规定的时间窗内服务一组顾客,以获得最大的总收益。提出了一类改进蚁群算法,用以求解该问题。为了提高解构造质量与效率,使用一种快速的方法来确定动态候选链表,并且利用串行法和贪婪法构造解。与迭代局部搜索相比,所提算法能够在12s内得到更好的解。

关键词 启发式算法,团队定向问题,蚁群优化,时间窗
中图法分类号 TP18 **文献标识码** A

Improved Ant Colony Optimization Approach for the Team Orienteering Problem with Time Windows

KE Liang-jun Zhang He SHANG Ke FENG Zu-ren
(State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract Team orienteering problem with time windows is one of the most important logistic distribution routing optimization problems. The optimization objective of this problem is to plan the optimal feasible routes in which the total reward is maximized while a set of customers can be served in the given time windows. An improved ant colony optimization approach was developed to deal with this problem. In order to construct better and faster solutions, the proposed algorithm uses a fast method to determine dynamic candidate list. Moreover, it adopts the sequence method and the greedy method to construct solutions. Compared with iterated local search, the proposed algorithm can obtain better results within 12 seconds.

Keywords Heuristic algorithm, Team orienteering problem, Ant colony optimization, Time windows

1 引言

近年来,团队定向问题受到了许多国内外研究者的关注,成为物流配送优化研究的热点之一[1,2]。在该问题中,一组车辆需要服务一定数量的点(顾客)。每条车辆的路径必须在规定的时间内从起点出发到达终点。一旦某条路径经过一个点,它将获得该点对应的收益,其他路径即使经过该点也不会获得该点对应的收益。团队定向问题的优化目标是使得总收益最大化。文献[2]很好地总结了与该问题相关的现有精确算法、启发式算法和应用等方面的研究成果。

在实际应用中,顾客往往需要在一定时间范围内得到服务,因此有必要研究带时间窗的团队定向问题。本文提出一类求解该问题的蚁群算法(ACO)。利用一种快速的方法来确定每一步解构造时可选点的候选链表;利用串行法和贪婪法构造解。实验表明,这些措施有助于提高解构造质量和计算效率。迭代局部搜索是一类高效的求解带时间窗的团队定向问题的算法[3]。实验结果表明,与迭代局部搜索相比,所提算法能得到更好的解。

2 问题描述

给定一个完全图 $G=(V,E)$,其中 $V=\{1,\cdots,n\}$ 是点(顾客)集,$E=\{(i,j)\mid i,j\in V\}$ 是边集。每个点 i 对应一个收益 r_i、一个服务或访问时间 T_i 和一个时间窗 $[O_i,C_i]$。起点为点1,终点为点 n,c_{ij} 为经过 E 中的边 (i,j) 所需要的时间。因此,带时间窗的团队定向问题即找出 m 条从点1出发到点 n 终止的路径,使得所经过点的总收益最大化。每个点最多只能经过一次。对于每条路径,访问相应的点所用的总时间不能超过预先规定的时间限制 $T_{\max}$。

如果路径 k 经过点 i,则 $y_{ik}=1(i=1,\cdots,n;k=1,\cdots,m)$,否则 $y_{ik}=0$;如果路径 k 经过边 (i,j),则 $x_{ijk}=1(i,j=1,\cdots,n;k=1,\cdots,m)$,否则 $x_{ijk}=0$;s_{ik} 为第 k 条路径中服务顶点 i 的开始时间。带时间窗的团队定向问题的数学描述为[3]

$$\max \sum_{i=2}^{n-1}\sum_{k=1}^{m} r_i y_{ik} \tag{1}$$

$$\text{subject to } \sum_{j=2}^{n}\sum_{k=1}^{m} x_{1jk}=\sum_{i=1}^{n-1}\sum_{k=1}^{m} x_{ink}=m \tag{2}$$

$$\sum_{i=1}^{n-1} x_{ijk}=\sum_{i=2}^{n} x_{jik}=2y_{jk}\ (j=2,\cdots,n-1;k=1,\cdots,m) \tag{3}$$

$$\sum_{k=1}^{m} y_{ik}\leqslant 1(i=2,\cdots,n-1) \tag{4}$$

$$\sum_{i=1}^{n-1}(T_i y_{ik}+\sum_{j=2}^{n} c_{ij}x_{ijk})\leqslant T_{\max}(k=1,\cdots,m) \tag{5}$$

$$O_i\leqslant s_{ik}\leqslant C_i(1\leqslant i<j\leqslant n;k=1,\cdots,m) \tag{6}$$

$$x_{ijk}\in\{0,1\},(1\leqslant i<j\leqslant n;k=1,\cdots,m) \tag{7}$$

到稿日期:2011-05-01 返修日期:2011-08-19 本文受国家重点基础研究发展计划(2007CB311000),国家自然科学基金(60905044),教育部博士点基金(20090201120042),博士后特别基金(201003675)资助。

柯良军(1976—),男,博士后,主要研究方向为优化理论及其应用、模式识别,E-mail:keljxjtu@xjtu.edu.cn。

$y_{1k}=y_{nk}=1, y_{ik}\in\{0,1\}(i=2,\cdots,n-1;k=1,\cdots,m)$ (8)

其中，约束(2)规定每条路径必须从起点 1 出发终止于 n，约束(3) 规定路径的连通性，约束(4) 规定除了点 1 和点 n 之外其他点最多只能经过一次，约束(5) 描述了时间约束。约束(6)描述了服务起始时刻的时间窗，约束(7,8) 保证了变量取值必须为整数。很显然，如果去掉约束(6)，带时间窗的团队定向问题就变成了团队定向问题。

3 本文提出的算法

蚁群算法是一类应用广泛的智能优化算法[4,5]。文献[6]中提出了一类求解无时间窗的团队定向问题。本文提出的算法与文献[6]中的蚁群算法的主要区别在于：(1)由于时间窗约束，它使用不同的启发信息；(2)为了提高搜索性能，它提供了一种有效的方法来构造候选链表；(3)利用候选链表，提出串行法和贪婪法，用以构造解。

算法的主要流程是：在每一代，每一个蚂蚁构造一个解；然后进行信息素的更新，算法在终止条件满足时停止。这里的终止条件采用最大运行时间。

3.1 启发信息的定义

假设第 k 条(未完成)路径当前顶点是 i，C 包含所有的候选可行顶点。假设该路径的已用时间为 L_{ki}，对于点 $j\in C$，如果点 j 的服务起始时间超出了时间窗(即 $L_{ki}+c_{ij}>C_j$)，或者该点不满足时间约束(5)，则该点不可行。不失一般性，假设点 j 可行。为了能经过边(i,j)，所用时间 $t(i,j)$为

$$t(i,j)=\begin{cases}c_{ij}+T_j, & \text{if } C_j>L_{ki}+c_{ij}>O_j\\ O_j+T_j-L_{ki}, & \text{if } O_j\geqslant L_{ki}+c_{ij}\end{cases} \tag{9}$$

由于带时间窗的团队定向问题的优化目标是在规定的时间内获得最大的总收益，蚂蚁将更偏好于那些有较高收益和较少时间消耗的点。因此，将启发信息定义为

$$\eta(i,j)=\begin{cases}g_{ij}=\dfrac{r_j}{(c_{ij}+T_j)^\gamma}, & \text{if } C_j>L_{ki}+c_{ij}>O_j\\ h_{ij}=\dfrac{r_j}{(O_j+T_j-L_{ki})^\gamma}, & \text{if } O_j\geqslant L_{ki}+c_{ij}\end{cases} \tag{10}$$

式中，γ 是参数。

因为启发信息 $\eta(i,j)$取决于 L_{ki}，所以当边(i,j)属于不同的路径时，$\eta(i,j)$可能会发生改变。注意到 h_{ij} 也取决于 L_{ki}。然而，对于任意可行点 $j\in C$，当构造第 k 条路径时，L_{ki} 是一样的。记 $h_j=r_j/(O_j+T_j)^\gamma$，则 g_{ij}，h_j 与 L_{ki} 无关。因此，如果静态保存 $G=(g_{ij})_{i,j\in J}$ 和 $H=(h_j)_{j\in J}$，就能提高计算启发信息的效率。而对于每两个顶点 $u,v\in C$，当且仅当 $h_{iu}\geqslant h_{iv}$ 时，$h_u\geqslant h_v$。矩阵 G 和向量 H 将在确定候选链表中起到重要作用。

3.2 确定候选链表

在算法运行过程中，依据启发信息，对可行点进行排序，将较好的可行点记录于候选链表中。如果直接排序，则在最坏的情况下计算复杂度是 $O(n\log(n))$，导致方法很费时。下面给出快速方法。

引入矩阵 $\bar{G}=(\bar{g}_{ij})_{i,j\in J}$ 和 $\bar{H}=(\bar{h}_j)_{j\in J}$，它们的定义如下：$\bar{g}_{ij}$ 对应于 g_{ij} 在 $\{g_{ij}\mid i,j=1,\cdots,n\}$ 中依大小排序得到的序号，$\bar{h}_j$ 对应于 h_j 在 $\{h_j\mid j=1,\cdots,n\}$ 中依大小排序得到的序号。不妨假设 C 的基数大于 $2N_d$，其中 N_d 是候选链表的大小。可通过以下步骤构造候选链表。

第 1 步　根据 $\bar{G}$ 的第 j 列，从 C 中选择前 N_d 点，将它们存于 I 中；

第 2 步　根据向量 $\bar{H}$，从 C 中选择前 N_d 点，将它们存于 J 中；

第 3 步　按照启发信息递减的顺序，从 $I\cup J$ 中选择前 N_d 点。

这样就能从集合 C 中确定 N_d 个具有较大启发信息的可行点。由于计算复杂度为 $O(N_d\log(N_d))$，因此用这种方法来确定候选链表是很有效的。

3.3 解构造

3.3.1 串行法

它利用候选链表以串行方式依次构造 m 条路径。在构造第 k 条路径时，蚂蚁从起点选择一个点，直到没有可行点，完成一条路径。假设第 j 步时可行点集为 C_j，$B_j\subseteq C_j$ 是当前候选链表，依概率从 B_j 中选择一个顶点 V_{j+1}，即

$$p(v_{j+1}=v\mid\tau)=\begin{cases}\dfrac{\tau(u,v)^\alpha\cdot\eta(u,v)^\beta}{\sum\limits_{w\in B_j}\tau(u,w)^\alpha\cdot\eta(u,w)^\beta}, & \text{if } v\in B_j\\ 0, & \text{else}\end{cases} \tag{11}$$

式中，α 和 β 是参数，其控制信息素和启发信息的相对重要性。

3.3.2 贪婪法

它以贪婪方式构造路径。首先每只蚂蚁按照式(11)分别构造出一条路径，从这些路径中选取最优路径作为第一条路径；然后，用类似的方法构造其他的 $m-1$ 条路径。

3.3.3 贪婪法和串行法的应用

前期实验结果表明，贪婪法能较快地构造出很好的解，但是它可能会导致算法很快陷入一个局部最优解。基于以上考虑，本算法中采用如下方式使用贪婪法和串行法：在第一阶段使用贪婪法，如果迭代 50 次仍不能找到新的优解，就使用串行法进行解构造。

3.4 信息素的更新

在每只蚂蚁都构造完一个解之后，采用极大极小蚂蚁系统(MMAS)[7]所给出的方法来更新信息素。对于每个边(u,v)，其信息素值更新为

$$\tau(u,v)^{l+1}=\rho\tau(u,v)^l+\Delta\tau(u,v) \tag{12}$$

$$\text{如果 } \tau(u,v)^{l+1}<\tau_{\min},\text{则 } \tau(u,v)^{l+1}=\tau_{\min} \tag{13}$$

$$\text{如果 } \tau(u,v)^{l+1}>\tau_{\max},\text{则 } \tau(u,v)^{l+1}=\tau_{\max} \tag{14}$$

式中，ρ 为信息素保持率($0\leqslant\rho<1$)，$\tau(u,v)^l$ 是边(u,v)在第 l 代时的信息素值。如果最优蚂蚁经过边(u,v)，则 $\Delta\tau(u,v)$为 $F(s_{\text{best}})$，否则 $\Delta\tau(u,v)=0$。$F(x)$为解对应的总收益与所有点的总收益之比，即

$$F(x)=\sum_{i=2}^{n-1}\sum_{k=1}^{m}r_i y_{ik}\Big/\sum_{i=2}^{n-1}r_i \tag{15}$$

s_{best}既可是本次迭代中最优解 s_{ib}，也可是全局最优解 s_{gb}。本算法中，利用如下混合策略来调度 s_{gb} 和 s_{ib}：每隔 5 代，用 s_{gb}

来更新信息素，而在其他代，利用 s_{ib} 更新信息素。τ_{max} 和 τ_{min} 分别是信息素上、下界，设置为

$$\tau_{max}=\frac{F(s_{gb})}{(1-\rho)} \tag{16}$$

$$\tau_{min}=(1-\sqrt[n]{P_{best}})/((\frac{n}{2}-1)\sqrt[n]{P_{best}})\tau_{max} \tag{17}$$

式中，P_{best} 是参数，通常取值为 0.05。

4 实验分析

本文算法用 Visual C++语言实现，测试计算机的 CPU 是奔腾 4，其主频是 3GHz。采用文献[3]中的 3 个算例进行实验分析，每个算例测试 30 次。当路径总数是 3 时，最大运行时间是 9s；当路径总数是 4 时，最大运行时间是 12s。实验表明，算法能在时间限制内很好地收敛。

在前期实验中，利用统计法选取参数如下：蚂蚁的数目设为 20，$\alpha=1$，$\beta=1$，$\rho=0.98$，$P_{best}=0.05$，$\gamma=3$，$N_d=10$。

4.1 候选链表和贪婪法的作用分析

为分析候选链表和贪婪法的作用，将本算法应用于路径数 $m=4$ 的算例 pr20 中。

首先比较候选链表大小 N_d 对算法的影响。实验中，考虑 N_d 的取值为 1，10，20 或 50。图 1 以盒图(boxplot)方式给出了不同取值下得到的总收益。注意到，当 N_d 为 10 时，均值最大，算法性能最好。随着 N_d 增大，解质量下降。而当 $N_d=1$ 时，由于算法以贪婪方式构造解，导致算法性能差。

其次，算法最大运行时间固定，N_d 对迭代次数也有影响。图 2 以柱状图方式(barplot)给出了 N_d 对迭代次数的影响，可见随着 N_d 增加，迭代次数下降。

最后，分析贪婪法对算法性能的影响。图 3 给出了每次迭代的平均总收益(图中给出了 1000 次迭代之前的结果)。由图可见，贪婪法能有效改善算法性能。

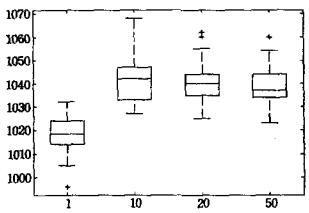

图 1 候选链表对目标函数值影响的实验结果

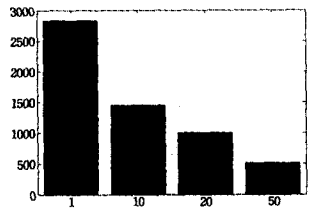

图 2 候选链表对迭代次数影响的实验结果

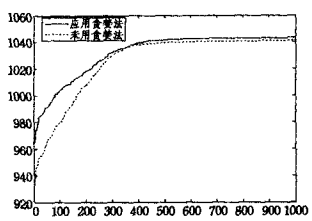

图 3 贪婪法对算法性能影响的实验结果

4.2 与迭代局部搜索(ILS)[3]的比较

表 1 和表 2 分别给出了迭代局部搜索和本文算法在路径数为 3 和 4 时的实验结果。在每个表中，分别给出了算例的名称、ILS 的最佳结果、本文算法的最佳结果和平均结果。

由表 1 和表 2 可见，本文算法能得到更好的最佳结果；而且，除了路径数为 3 的 pr20 一个算例之外，所提出算法得到的均值对优于 ILS 的最佳结果。实验结果表明了本文算法的有效性。

表 1 路径数目是 3 时与迭代局部搜索(ILS)的比较

算例	ILS	所提出算法		
		最佳	均值	均方差
pr02	899	922	894.6	11.0
pr03	946	999	973.8	7.3
pr04	1195	1265	1227.4	17.7

表 2 路径数目是 4 时与迭代局部搜索(ILS)的比较

算例	ILS	所提出算法		
		最佳	均值	均方差
pr02	1014	1068	1041.7	10.3
pr03	1162	1220	1164.3	19.8
pr04	1452	1553	1471.9	23.9

由于 ILS 的测试平台有所不同，因此难以直接比较计算时间。不过，以 pr20 为例，ILS 在主频为 2.5GHz 的主机上耗时为 11.6s。因此，所提出算法计算耗时也相差不大。

结束语 针对具有时间窗约束团队定向问题，本文提出了一类改进蚁群算法。为了更加快速有效地构造解，提出了一种快速的方法来建立候选链表。此外，在搜索的第一阶段用贪婪法来构造解，然后用串行法。实验结果表明，它们有助于改善算法性能。实验结果表明，与迭代局部搜索相比，所提出算法可以在 12s 内找到更好的解。在后续研究中，拟将所提算法推广到其他有时间窗约束的优化问题，如带时间窗的车辆路径问题。

参 考 文 献

[1] Chao I M, Golden B, Wasil E A. The team orienteering problem [J]. European Journal Operational Research, 1996, 88: 464-474

[2] Vansteenwegen P, Souffriau W, Oudheusden D V. The orienteering problem: A survey [J]. European Journal of Operational Research, 2011, 209(1): 1-10

[3] Vansteenwegen P, Souffriau W, Berghe G V, et al. Iterated local search for the team orienteering problem with time windows [J]. Computers and Operations Research, 2009, 36: 3281-3290

[4] Dorigo M, Gambardella L. Ant colony system: A cooperative learning approach to the travelling salesman problem [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53-66

[5] Dorigo M, Stützle T. Ant colony optimization [M]. MA: MIT Press, 2004

[6] Ke L, Archetti C, Feng Z. Ants can solve the team orienteering problem [J]. Computers and Industrial Engineering, 2008, 54 (3): 648-665

[7] Stützle T, Hoos H H. Max-min ant system[J]. Future Generation Computer Systems, 2000, 16: 889-914