文章编号:1671-4598(2018)11-0228-05 DOI:10.16526/j.cnki.11—4762/tp.2018.11.050 中图分类号:TP273 文献标识码:A

共享适应度粒子群在双机 ETV 中的应用

丁 芳,宋小静
(中国民航大学 电子信息与自动化学院,天津 300300)

摘要:针对机场货运区双机双货位升降式转运车(ETV)并行工作时任务链生成困难的问题,提出一种基于 ETV 载物台的任务链生成算法;该算法结构清晰,且易于计算总运行时间;针对粒子群算法在计算双机 ETV 最优任务序列时易出现早熟问题,提出改进的共享适应度粒子群算法。该算法在混沌粒子群算法的基础上融合共享适应度的思想,当混沌粒子群算法的全局最优解趋于稳定时,选取百分之二十的粒子留守共享半径内,其他粒子重新初始化并继续迭代;实验仿真结果表明和标准粒子群算法、混沌优化粒子群算法相比,共享适应度粒子群算法可以有效地避免早熟,全局寻优能力更强,得到的结果更优且更稳定。

关键词:ETV;调度;任务链;粒子群;共享适应度

Application of Shared Fitness Particle Swarm in Double ETV System

Ding Fang, Song Xiaojing
(College of Electronic Information and Automation, China Civil Aviation University, Tianjin 300300, China)

Abstract: Aiming at the difficulty of task chain generation when two EVTs work in parallel in the airport cargo area, a task chain generation algorithm based on ETV platform is proposed. This algorithm has a clear structure and easy to calculate the total running time. Aiming at the problem of premature maturity of particle swarm optimization algorithm, an improved shared fitness particle swarm optimization algorithm is proposed. Twenty percent of the particles are selected to be in the left-behind Shared radius, and other particles are re-initialized and iterated, when the global optimal solution of the algorithm tends to be stable. The simulation shows that, compared with the standard particle swarm algorithm and chaotic particle swarm algorithm, the Shared fitness particle swarm algorithm can effectively avoid the precocity, and the global optimization ability is stronger. The solution is better and more stable.

Keywords: ETV; scheduling; task chain; particle swarm; shared fitness

0 引言

升降式转运车(Elevating Transfer Vehicle, ETV)是机场货运区运输集装货物的转运工具。许多大型机场在立体仓库内配置了两台甚至更多的 ETV 同时作业。这些 ETV 运行在同一巷道内,不可避免的会出现碰撞。另外每个 ETV 通常拥有两个货位,立体仓库内有两排货架,每个货架有两个货位,这些情况都显著提升了调度过程的复杂度。所以研究双机双货位 ETV 控制系统对提高机场货运区转运效率和减少 ETV 磨损有着重要的意义。

近年来一些学者对双机 ETV 调度算法进行了研究。宋宇博[1]等分析航空货站 AS/RS 多端口出入库的作业方式具有并发性和分布性特点,调度过程具有高密度和高柔性的特点,提出了一种两阶段禁忌搜索算法,并研究了 ETV 防冲突避让算法,结果表明,此算法取得了良好的效果。季琼[2]等使用分配法对专家系统进行改进,建立了双 ETV 任务分配调度专家系统的知识库,提升了机场货运站的作业效率。YB[3]等使用带约束的优化模型分析双机 ETV 调度问题。以上方法均是基于任务集并结合 ETV 载物台的状态生成任务链,该过程建模复杂且不易实现。论文针对这个问题提出一种基于 ETV 载物台状态,结合下一任务属性生成任务链的算法。该算法较基于任务的模型结构更加清晰,编程简单易行。

标准粒子群算法易陷入局部最优,常见的优化算法采用调节权重、增加变异项等[4-6]。实验仿真发现,这些优化算法在处理双机 ETV 调度问题时虽然有一定的效果,但是跳出局部最优的能力依然较差且出现早熟现象。为了解决这个问题,引入共享适应度函数,改进了粒子群算法。

1 双机 ETV 复合调度建模和粒子群算法

ETV 所在的立体仓库结构如图 1 所示。

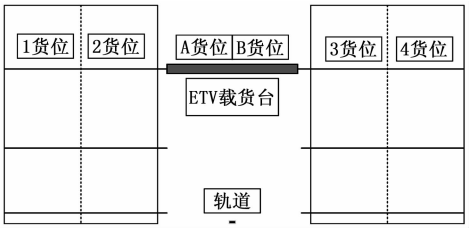

图 1 机场立体仓库结构示意图

收稿日期:2017-05-23; **修回日期**:2018-10-31。
基金项目:中央高校基本科研业务费项目中国民航大学资助专项(3122017003)。
作者简介:丁 芳(1960-),女,上海人,工学硕士,副教授,主要从事智能控制、检测方向方向的研究。

图中看出立体仓库分为两排，每排有 n 层 m 列货位，ETV 载物台有 A、B 两个货位。每个货位可装载一个 10 号集装器（Unit Load Device，ULD），两个货位合并可装载一个 20 号集装器。为了描述方便，约定称 10 号集装器为小箱，称 20 号集装器为大箱，并且约定 2、3 号货位所在排为内侧货位，1、4 号货位所在排为外侧货位。

机场货运区立体货架地址采用排－层－列的顺序编码。例如国内某机场拥有 4 排 5 层 45 列货架。为了计算方便，将三维编码按照公式（1）映射为一维编码。

$$s = 5(x-1) + y + 20(z-1) \tag{1}$$

式中，s 为地址，x 为排，y 为层，z 为列。例如第 1 排第一层第一列的地址编码为 1，以此类推。

1.1　任务集和任务链分析

任务的合集称之为任务集，任务的属性包括出、入、倒、盘库等。每个出/入库任务选择最近的出/入口作为其目的/源地址，每个倒/盘库任务将下一个节点的地址作为其目的地址。每个任务映射为一个向量即 $C(i)$ = (srcadd (i)，dstadd (i)，size (i))，向量内元素依次为：源地址、目的地址、货物大小。

任务集内的任务按照一定的先后顺序组成一个任务序列，这些任务序列的源地址、目的地址首尾相连组成一个任务链。任务链节点是一个向量即 $L(i)$ = (add (i)，dis (i))，向量内元素依次为：地址、x 轴位移，其中任务链节点地址仅包含 y、z 两个方向，x 轴位移根据每个任务的源/目的地址和货物大小综合计算。任务链描述的是 ETV 载物台在立体仓库内运动的轨迹。例如任务链两个相邻节点 (add_1，－1)、 (add_2，1)，描述的是 ETV 载物台在地址 add_1 处获取第三排货位的 ULD，然后运动到 add_2 地址处向第三排卸载此 ULD。

许多研究人员从任务的角度出发研究 ETV 的调度问题，分析复杂且易出错。从任务链的角度出发将调度问题简化为一个具有马尔科夫性的过程，即：任务链的下一个节点状态仅和当前任务状态及下一个任务有关，而和任务链之前的节点无关。另外基于任务链的模型仅计算每个节点 x 轴方向运行时间和相邻两节点运行时间即可计算出总的运行时间，较基于任务的模型结构更加清晰。任务链生成步骤如图 2 所示。

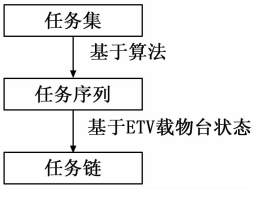

图 2　生成任务链步骤

1.2　按照任务序列生成任务链

ETV 载物台有两个货位，理论上可以同时载入两个小箱，但是如果两个小箱目的地址有竞争，ETV 会产生死锁。解除死锁的方法主要有 4 种：①撤销全部有关任务；②依次撤销最后加入的任务，直到死锁消失；③强制剥夺陷入死锁进程的资源，直到死锁消失；④调用更多的资源分配给死锁进程。具体到 ETV 调度问题有两种策略避免死锁：①撤销最后一个任务；②寻找中转货位。文献[7]证明多数情况下采用中转货位策略花费时间长于撤销最后一个任务策略，为了减少 ETV 的磨损且减少运行时间，论文采用第二种 ETV 死锁避免策略生成任务链。

按照任务序列顺序，以 ETV 的两个载物台货位状态为条件并加入死锁避免策略，生成任务链。假设 ETV 载物台的两个货位用 m_1、m_2 表示，分以下几种情况：

1）m_1、m_2 均为空。载入下一个任务。

①如果下一任务货物类型为内侧小箱，将任务信息赋值给离此地址最近的载物台。然后将任务链的下一节点赋值为载入任务的源地址和货物类型。此时载物台有一个小箱，任务链增加了一个节点。

②如果下一任务为外侧小箱且目的地址为同侧，将任务信息赋值给与任务货位最近的载物台货位，将远离任务货位的载物台货位的源、目的地址均赋值为与任务货物相邻的内侧货位的地址。然后将任务链的下一节点赋值为载入任务的源地址和货物类型。此时载物台有两个小箱，任务链增加了两个节点。

③如果下一任务为外侧小箱且目的地址为对侧，按照距离下一任务的目的地址最节省时间且在对侧的空闲货位为中转货位；任务完成后，载物台上只剩中转货物，源地址为中转货位地址，目的地址为中转货位的原始地址。任务链增加了 3 个节点。第一个节点的地址为中转货位地址，货物类型为 1；第二个节点的地址为任务的目的地址，货物类型为 1；第三个节点的地址为中转货位地址，货位类型为 1。

③如果下一任务链货物类型为大箱，将任务信息赋值给载物台所有货位。将任务链的下一节点地址赋值为载入任务的源地址，货物类型为 2。

最后更新立体货架存货信息。

2）m_1、m_2 有一个为空。

如果是下一任务货物类型为大箱，按照死锁避免策略应当首先清空载物台再进行装载，装载完成时，载物台上只有一个大箱，载物台两货位信息为大箱的源地址、目的地址和货物类型。按顺序增加两个任务链节点：1、当前任务的目的地址，货物类型为 1；2、下一任务的源地址，货物类型为 2。最后更新立体货架存货信息。

如果不是大箱，分几种情况分析：

①m_1 有货，目的地址为同侧，直接载入下一任务。当下一任务源地址和 m_1 同侧时，将 m_1 赋值给 m_2，将下一任务的信息赋值给 m_1。此时载物台上有两个货物。当下一任务源地址和 m_1 异侧时，将下一任务信息赋值给 m_2。

任务链增加一个：节点为下一任务的源地址，货物类

型为 1。

②m_1 有货，目的地址为异侧。如果下一任务源地址为 m_1 同侧，将 m_1 赋值给 m_2，下一任务信息赋值给 m_1，此时载物台上有两个货物。如果下一任务源地址为 m_1 异侧目的地址 m_1 异侧，将下一任务信息赋值给 m_2。

任务链增加一个：节点为下一任务的源地址，货物类型为 1。

如果下一任务源地址为 m_1 异侧目的地址 m_1 同侧，此时产生死锁。使用撤销策略避免死锁，首先完成当前任务，然后装载下一任务，完成时载物台有一个 m_2 货物，m_2 属性为下一任务属性，

任务链增加两个：第 1 节点为当前任务的目的地址，货物类型为 1；第二节点为下一任务的源地址。

③m_2 有货，分析方法和第二类 m_1 有货相似。

3）m_1、m_2 均有货。

如果 ETV 载物台两货物均为小箱，选择离 m_1、m_2 目的地址耗时最短的货位为下一任务链节点。①下一任务是卸载 m_1，将 m_2 赋值给 m_1，清空 m_2，增加一个任务链节点：m_1 目的地址。②下一任务是卸载 m_2，将 m_1 赋值给 m_2，清空 m_1，增加一个任务链节点：m_2 目的地址。完成后 m_1、m_2 有一个为空。

如果 ETV 载物台为一个大箱，下一任务是卸载 m_1 和 m_2，任务完成后清空 m_1 和 m_2，任务链增加一个节点：地址为 m_1 或 m_2 的目的地址，货物类型为 2。

1.3 使用粒子群算法求任务序列

国内某机场某立体仓库内拥有一个巷道两个 *ETV*，双机 *ETV* 调度问题属于并行调度问题（*Parallel Machine Shop Scheduling Problem*，*PMSP*），*PMSP* 是 *NP* 完全问题[8]。随着启发算法的深入研究，*NP* 完全问题得到了很好的解决。其中粒子群算法结构简单易行得到了广泛的应用。

双机 *ETV* 调度问题是带有约束条件的优化问题，它的适应度函数为式（2）：

$$p_i = \max(T_{1i}, T_{2i}) \tag{2}$$

式中，T_{1i}、T_{2i}分别表示 1 号 *ETV* 和 2 号 *ETV* 执行任务链所用时间。

约束条件如式（3）所示：

$$\begin{cases} 1 \leqslant ETV^1 \leqslant 40 \\ 5 \leqslant ETV^2 \leqslant 45 \\ |L^1(t) - L^2(t)| \geqslant 4 \\ L^1(i) \in Task \\ L^2(i) \in Task \\ task^1(i) \notin task^2, task^2(j) \notin task^1 \end{cases} \tag{3}$$

式（3）中，第一行约束条件保证一号 *ETV* 运行的区间在第 1～40 列，第二行约束条件保证二号 *ETV* 运行的区间在第 5～45 列，第三行约束条件保证相同时刻两 *ETV* 之间的距离大于等于 4 列，第四行和第五行保证两个任务链均属于任务集，第六行保证两个子任务集内任意元素不属于对方任务集。

1.4 双 ETV 调度问题粒子编码

双机 *ETV* 调度要解决的是任务分配问题，将任务、*ETV* 序号映射为粒子的位置，这个过程叫做编码。相对应的，将粒子位置映射为任务、*ETV* 序号的过程称之为解码。

设有 40 个出入库任务，则一个粒子需要 40 个维数，将粒子的每个维数和每个任务一一对应，根据粒子每个维度的坐标确定任务的序列。首先将粒子每个维度的坐标值按照从大到小的顺序排列。将坐标值大于 0 的维度对应的任务分配给 1 号 *ETV*，将坐标值小于 0 的维度对应的任务分配给 2 号 *ETV*。并且每个 *ETV* 分配的任务执行顺序为粒子坐标值的顺序。例如 10 个粒子分配情况如表 1 所示。

表 1 粒子编码示例

任务	1	2	3	4	5	6	7	8	9	10
坐标	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
取值	1.89	−1.31	−0.97	0.25	1.52	−0.57	1.32	1.96	−2.09	−1.49

表中，1 号 *ETV* 执行的任务序列为 8、1、5、7、4；2 号 *ETV* 执行的任务序列为 6、3、2、10、9。

1.5 标准粒子群算法

在一个 N 维空间中有 M 个粒子，每个粒子的位置记为 $x_i=(x_1, x_2, \cdots, x_n)$，速度记为 $v_i=(v_1, v_2, \cdots, v_n)$。在基本粒子群算法中，计算每个粒子的适应度，如果第 i 个粒子在某次迭代中适应度比以前更优，则更新第 i 个粒子的最优值并记做 p_{id}；选出这些粒子的最优值记做全局最优值并记做 p_{ig}。按照公式（4）进行位置的迭代。

$$\begin{cases} v_i^{(n+1)} = wv_i^n + c_1 r_1 (p_{id}^{(n)} - x_i^n) + c_2 r_2 (p_{ig}^{(n)} - x_i^n) \\ x_i^{(n+1)} = x_i^n + v_i^{n+1} \end{cases} \tag{4}$$

式中，w 为惯性系数，w 等于 0.9；c_1、c_2 分别为认知系数和社会系数，c_1、c_2 均等于 2；P_{id}为单粒子最优值；P_{ig}为所有粒子的最优值；r_1 和 r_2 属于 U（0，1）均匀分布；

继续迭代，直至满足停止条件，全局最优解 P_{ig}即为模型最优解，对应的粒子位置 x_{ig}即为最优位置。

1.6 混沌粒子群算法

许多改进的粒子群算法利用复杂的函数调节速度迭代算式，但是仿真结果发现在求解本文问题时容易陷入局部最优。Khare[9]等提出一种混沌迭代粒子群算法，王[10]等增加了一种变异因子。这些算法均提高了跳出局部最优的能力。受此启发对认知系数和全局系数进行非线性优化。为了提高迭代后期样本的多样性，利用混沌算子可以不重复的遍历吸引域内的所有点的特性，引入 Tent 混沌算子进行变异。

混沌优化粒子群算法中，位置更新算式如式 5 所示。

$$\begin{cases} v_i^{(n+1)} = wv_i^n + c_1 r_1 (p_{id}^{(n)} - x_i^n) + c_2 r_2 (p_{ig}^{(n)} \times (r^n)^s - x_i^n) \\ x_i^{(n+1)} = x_i^n + v_i^{n+1} \\ c_1^n = c_{1end} + (c0_{1start} - c_{1end}) \tan\left(k_1 \left(1 - \left(\frac{n}{n_{\max}}\right)^k\right)\right) \\ c_2^n = c_{2end} + (c_{2start} - c_{2end}) \tan\left(k_2 \left(1 - \left(\frac{n}{n_{\max}}\right)^k\right)\right) \end{cases} \tag{5}$$

式中，惯性权重 w 为 0.9；c_1、c_2 为非线性参数项，$c_{1\text{start}}$为 2.5，$c_{1\text{end}}$为 1、$c_{2\text{start}}$为 0.5、$c_{2\text{end}}$为 2.25；k_1 和 k_2 均为$\frac{\pi}{4}$，k 为 1；s 为自适应系数，当适应度在迭代时小于给定值的时候 $s=1$，其他时刻 $s=0$；r^n 为混沌变异项，混沌算子为 Tent 映射算子，如式（6）所示：

$$r^{n+1}=\begin{cases}10r^n/7 & 0\leqslant r^n<0.7\\ 10r^n(1-r^n)/3 & 0.7\leqslant r^n\leqslant 1\end{cases} \tag{6}$$

2　共享适应度改进粒子群算法

改进的粒子群算法虽然加快了收敛速度，而且在一定程度上提高了收敛效果，但是仍然易出现早熟现象。为了解决这个问题，论文在混沌粒子群算法的基础上，采用共享函数改进粒子群算法，提高算法的收敛效果。共享适应度算法来源于多峰函数的遗传算法求解，其思想是将某个小生境内的所有个体的适应度值除以一个和个体数目相关的函数值，用来鼓励较少个体物种的繁殖。论文借用这个思想在混沌粒子群算法的基础上，划定了一个共享半径，通过共享适应度的方式驱离重新初始化的粒子。

共享函数描述的是两个粒子之间的相互关系，它的值由此粒子和其他粒子之间的关系决定。增加了共享函数的粒子适应度为式（7）：

$$\overline{p_i}=\frac{p_i}{S_i} \tag{7}$$

式中，p_i 为原始适应度函数，S_i 为共享函数，$\overline{p_i}$ 为更新的适应度函数。

两个粒子位置向量差值的 1 范数记做两个粒子之间的距离，即公式（8）：

$$d_{ij}=\sum_{k=1}^{n}|x_i(k)-x_j(k)| \tag{8}$$

当粒子群陷入到局部最优时随机选择一部分粒子重新初始化，新旧两个粒子群相对独立。为了防止新粒子群回到原粒子群的局部最优附近，选取早熟区范围是距离全局最优解最近的 20% 的粒子，这些粒子中离全局最优解最远的粒子距离设为共享半径 d_s，当新粒子进入到共享半径时，它的适应度函数应当急剧增加，以促使其远离这个区域。基于以上思想设计共享适应度函数为公式 9：

$$\overline{p_i}=\frac{p_i}{d_i/Md_s} \tag{9}$$

式中，d_s 为共享半径，d_i 为粒子到原全局最优解的距离，M 是惩罚系数，为定值 100。

改进算法流程如下。

Step1：初始化粒子群。

Step2：在满足约束条件的基础上计算各个粒子的适应度，判断是否到达迭代次数的最大值，如果是则跳转到 Step6，如果否继续。

Step3：判断收敛条件，如果收敛说明陷入局部最优，跳转至步骤 5，如果否继续。

Step4：按照公式（2）更新位置和速度，然后跳转到 Step2。

Step5：按照 1.5 章所述算法确定共享半径，选出部分粒子重新初始化，并且计算这部分粒子的共享适应度并排序，判断所有粒子是否均在共享半径外。如果有粒子在共享半径内，继续初始化这些共享半径内的粒子，直到所有新粒子均在共享半径外，然后跳转到 Step2。

Setp6：迭代停止，得到最优解。

3　实验和结果

国内某机场 ETV 仓库拥有 7 个路侧 I/O 口和 6 个空侧 I/O 口，设路侧为入口、空侧为出口，则出入口分布如表 2 所示。

表 2　出入口位置

	位置						
入口	86	206	366	406	626	686	886
出口	151	311	531	591	791	851	

实验样本选用 20 个入库任务和 20 个出库任务，如表 3 所示。

表中，任务号格式为 X－X－X，以此表示为任务类型、

表 3　任务库

任务号	货位号	任务号	货位号	任务号	货位号	任务号	货位号
IN－1－1	810	IN－11－1	669	OUT－1－2	10	OUT－11－1	749
IN－2－1	130	IN－12－1	68	OUT－2－1	27	OUT－12－1	630
IN－3－2	328	IN－13－2	848	OUT－3－1	435	OUT－13－2	94
IN－4－1	248	IN－14－1	890	OUT－4－1	186	OUT－14－1	233
IN－5－1	266	IN－15－1	52	OUT－5－1	370	OUT－15－1	711
IN－6－1	147	IN－16－1	588	OUT－6－1	450	OUT－16－1	275
IN－7－1	487	IN－17－1	313	OUT－7－1	448	OUT－17－1	169
IN－8－1	570	IN－18－1	533	OUT－8－1	389	OUT－18－1	472
IN－9－2	194	IN－19－1	655	OUT－9－1	15	OUT－19－1	271
IN－10－1	667	IN－20－1	855	OUT－10－2	747	OUT－20－1	713

任务号、大小箱类型。其中大小箱类型分别表示‘1’为小箱和‘2’为大箱。

3.1 算法步骤

算法步骤如 Step1～5 所示。

Step1：选择距离最近的出/入口作为出/入库任务的目的/起始坐标生成只含有源/目的地址的任务集。

Step2：判断是否达到停止条件。如果是，跳转到 Setp5；如果否，继续。

Step3：按照标准 PSO 算法、混沌优化粒子群算法、共享适应度粒子群算法分别迭代生成新的编码顺序

Setp4：根据粒子群算法生成的编码顺序生成任务链，并按照公式（2）计算粒子群的适应度。然后跳转到 Step2。

Step5：迭代停止，得到最优解。

3.2 仿真结果和分析

3 种粒子群算法的最大迭代次数均为 200 次。粒子群规模均为 20 个。为了测试算法的有效性，分别运行 10 次并记录。10 次迭代结果分别如图 3～5 所示。

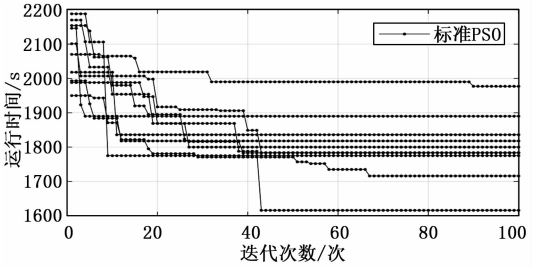

图 3 标准 PSO 迭代结果

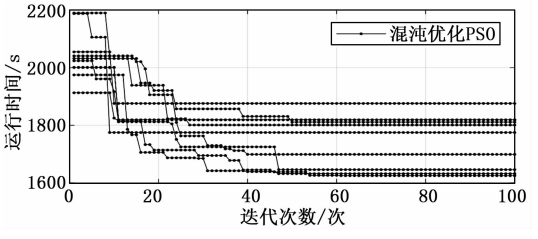

图 4 混沌优化 PSO 迭代结果

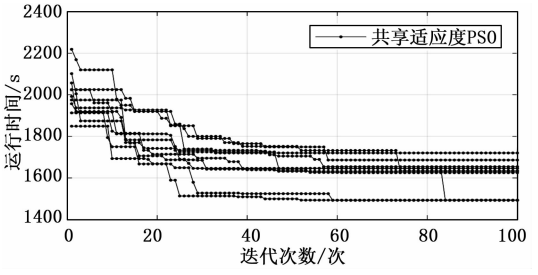

图 5 共享适应度 PSO 迭代结果

从图 3～5 可以看出，该样本有 3 个局部最优解区域，分别在 1 500 s、1 650 s、1 800 s 左右。标准粒子群算法易出现早熟现象。混沌粒子群算法虽然加快了运行速度而且在一定程度上提高了跳出局部最优的能力。从图 5 可以看出，在迭代后期有一些粒子到达了 1 500 s 左右区域，共享适应度粒子群算法在增加少量迭代次数的情况下有较强的跳出局部最优的能力，全局寻优能力更强。

计算 10 次任务链的平均时间如图 6 所示。

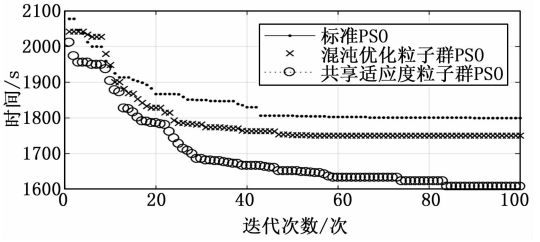

图 6 平均任务链时间对比

图中可以看出，共享适应度粒子群算法所得任务链时间最短。

使用公式（10）计算第 i 次迭代中 10 次结果的标准差，如图 7 所示。

$$\sigma=\sqrt{\frac{1}{10}\sum_{i=1}^{10}(x_i-\mu)} \tag{10}$$

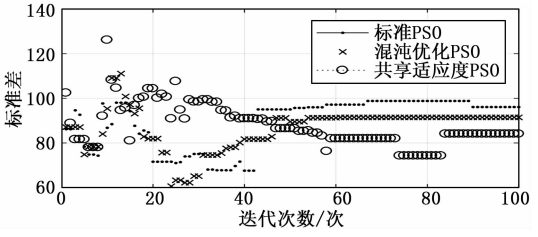

图 7 标准差对比

图中可以看出迭代初期标准 PSO 的标准差最低，共享适应度粒子群算法标准差波动最大。迭代后期混沌优化粒子群算法标准差最大，共享适应度粒子群算法标准差最小。结果说明迭代初期共享适应度粒子群算法频繁的执行粒子重初始化程序，使粒子跳出局部最优；迭代后期共享适应度粒子群算法所得结果更稳定。

图 8 是共享函数改进粒子群算法获得的最优序列的位置－时刻图。

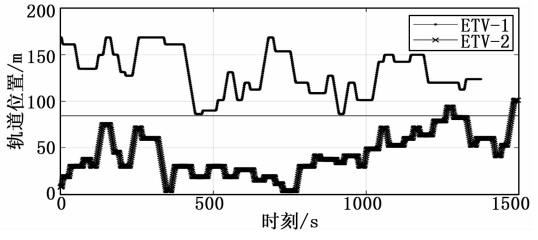

图 8 最优位置时刻图

图中可以看出，最优序列下两个 ETV 大部分时间在各自的半区运行，且运行轨迹较平滑，进一步验证了算法的有效性。

（下转第 247 页）