杭州電子科技大学

学　　院	管理学院
专　　业	管理科学与工程
小组成员 1	张丁鹏
小组成员 2	季煜卿
小组成员 3	冯晨翀
课程名称	智能计算及软件应用
完成日期	2024 年 6 月

作业车间调度问题的混合粒子群优化和模拟退火算法的介绍与实现

1.论文介绍

运输资源作业车间调度问题(JSPT)涉及确定机器调度、运输分配和车辆调度。虽然同时解决这些相互关联的问题可以提高制造系统的整体性能，但由于给问题增加了额外的决策和约束，它们的集成使调度问题变得更具挑战性。因此，制定适当的调度方法以获得满意的结果至关重要。

JSPT 是 np-hard 的，因为它扩展了两个 np-hard 问题:作业车间调度问题(JSP)和车辆调度问题，这类似于取货和交货问题。与 JSP 的情况一样，大多数关于 JSP 的研究都最小化了所有作业的最大完成时间。然而，在文献中，有一些作品最小化了退出时间，即处理和返回所有作业到 LU 所需的时间。尽管这两个指标紧密相连，因为它们都旨在最小化作业集的完成时间，但退出时间更现实和准确，因为只有在将作业交付给业务单元之后，它们才准备好发送给最终客户。

由于 JSPT 是 np-hard 的，所以精确的方法只能解决小型实例。因此，该论文提出了一种混合粒子群优化和模拟退火算法(PSOSA)，该算法可以在合理的计算时间内找到最优或近最优解。为了评估小型实例的 PSOSA 解，我们使用已有文献在最小化 Cmax 时通过求解 MILP 模型获得的解，以及在最小化 ET 时通过求解该模型的修改版本获得的解。此外，本文还设计了一个非常快速的下限程序，我们使用它来评估中型和大型实例的 PSOSA 解决方案。最后，进行了大量的计算实验，以显示所提出方法的有效性和效率，并为处理如此复杂问题的管理者提供一些见解。

该论文的主要贡献有两个方面:首先，提出了一种混合粒子群优化和模拟退火算法(PSOSA)，能够找到高质量的解决方案，并优于最先进的算法。第二，提供通过一个非常快速的下界过程，得到了 PSOSA 解的下界。此外，采用了已有文献的 MILP 配方，以最大限度地减少退出时间。

2.算法介绍

2.1 问题定义和描述

该论文所考虑的作业车间调度问题的主要组成部分是生产系统和运输系统。生产系统是一个传统的作业车间调度问题;因此，它包括一组独立的作业和一组机器。每个作业由一组有序的操作组成，每个操作必须在预定义的处理时间内在专用机器上不间断地处理。因此，每个作业都有自己独立的机器顺序。此外，每台机器一次只能处理一个操作，每个作业一次只能在一台机器上处理。运输系统由几辆相同的车辆组成，这些车辆以相同的恒定速度移动，每次可以进行一项工作。

因此，任何车辆都可以执行任何运输任务。由于布局是已知的，并且运输不是先发制人的，任何两个地点之间的旅行时间都是预先确定的。关于传输任务，之前已经解决了三个不同版本的问题。在“最简单”的方法中，不考虑逻辑单元，因此，作业只在机器之间传输。作业最初由机器处理它们的第一个操作，当它们的最后一个操作被处理时，它们就被认为完成了。

如图 1 所示，所有的作业和车辆最初都位于 LU。然后，每个作业由一辆车从逻辑单元(它进入生产系统的地方)运送到处理其第一次操作的机器。如果机器空闲，则立即开始加工;否则，作业将在机器输入缓冲区中等待。在任何一种情况下，一旦车辆到达机器，它就会放下工作，继续下一个任务。一旦一个作业的操作完成(例如作业 j 的操作 *i*)，车辆需要从机器加工操作 *i* 中拾取作业 j，并将其运送到机器加工操作 *i* + 1 中。每当一项操作完成时，该作业要么进入机器输出缓冲区，在那里等待车辆的到来，要么直接进入已经可用的车辆。无论哪种方式，机器都可以开始处理下一个操作。如果车辆在作业完成之前到达处理作业的机器，则必须等待取件。注意运输需要取货和送货。而后者总是涉及旅行，前者只要求车辆行驶(空)，如果车辆不在取货地点。在处理完作业的所有操作之后，作业将返回到 LU(它从这里离开生产系统)。

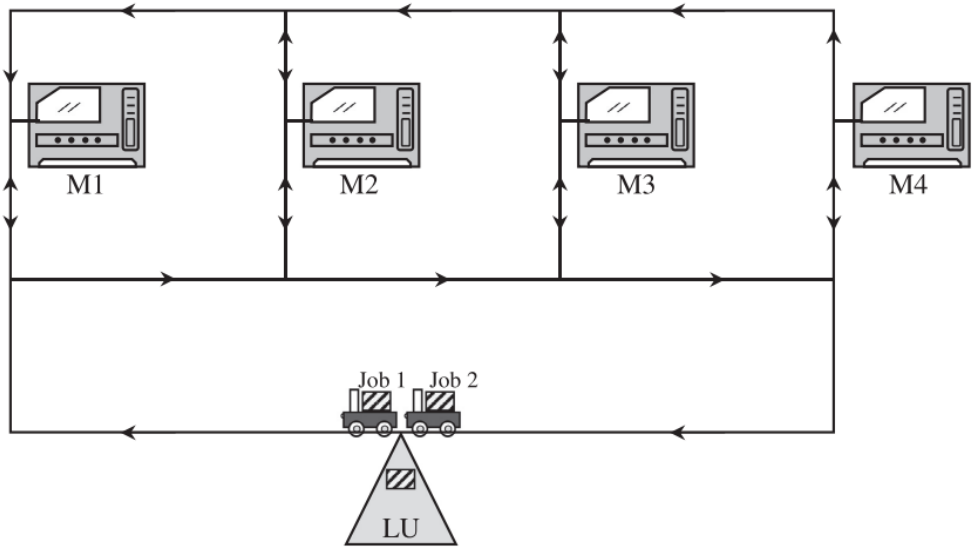

图 1 说明性示例：基于布局 1 的车间配置

2.2 模型描述

文章提出了 MILP 模型的修改版本，该模型(i)最小化退出时间 ET，(ii)考虑到工作返回到 LU。在给定的机器上进行一项操作，需要将该作业传送到相应的机器上。因此，除了将作业返回给 LU 所需的任务外，每个传输任务都与一个生产操作相关联。相关符号说明见表 1。

表 1 符号说明

M	机器集合，以 M 为索引;
J	作业集，以 J 和 l 为索引;
f,t	第一个和最后一个虚拟作业，每个作业由与机器数量相同的操作组成，分别是处理它们的机器的第一个和最后一个操作;
J_j	作业 j∈J∪ $\{f,t\}$的 n 个操作的集合，以 i 和 k 为索引
n_j+1	作业 j∈J 在处理时间为 0 的 LU 处处理的虚拟最后一次操作。
J_j'	作业 j∈J 的操作集合，包括假操作;即 J_j=J∪ $\{n_j+1\}$。
O_{ij}	作业 j∈J 的操作 i∈J_j;
M_{ij}	处理操作 Oij 的机器，i∈J_j,J∈J，其中 M(n_j+1)J 根据定义为 LU;
T_{ij}	传输任务，将作业 j 传递给机器 $M_{ij,}$ i∈j_j,j∈J;
P_{ij}	操作 O_{ij} 的处理时间，i∈J_j, j∈J, P(n_j+1)J=0;
$\mathcal{T}_{ij}^{kl}$	机器 M_{ij} 到机器 M_{kl} 的行程时间，i∈J_j，k∈J_l，j,l∈J;
A	可用车辆数量;
LN	一个足够大的正整数
决策变量	
w_{ij}^{kl}	如果在操作 O_{ij} 之后立即进行操作 O_{kl}，则取值为 1,否则取 0,k∈Jl，l∈J∪{t}， i∈J , J∈J∪{f};
x_{ij}^{kl}	二进制变量，如果运输任务 T_{kl} 在运输任务 T_{ij} 之后立即由同一车辆完成，则值为 1，否则为 0
u_{ij}	二进制变量，如果运输任务 T_{ij} 是车辆的第一项任务，则取 1，否则取 0
z_{ij}	二进制变量，如果传 输任务 T_{ij} 是车辆的最后一个任务，则取 1，否则取 0
辅助变量	
ET	上一项工作的离职时间
c_{ij}	O_{ij} 操作完成时间
v_{ij}	车辆到达 M_{ij} 机器的时间

Minimize ET （1）

$$ET \geq c_{(n_l+1)l}, \forall l \in J \quad (2)$$

$$\sum_{i<k\in J_l} w_{il}^{kl} + \sum_{j\in J\setminus\{l\}} \sum_{i\in J_j} w_{ij}^{kl} + \sum_{i\in J_f} w_{if}^{kl} = 1, \forall l \in J\cup\{t\}, k \in J_l \quad (3)$$

$$\sum_{i>k\in J_l} w_{kl}^{il} + \sum_{j\in J\setminus\{l\}} \sum_{i\in J_j} w_{kl}^{ij} + \sum_{i\in J_t} w_{kl}^{it} = 1, \forall l \in J\cup\{f\}, k \in J_l \quad (4)$$

$$\sum_{j\in J} \sum_{i\in J_j'} z_{ij} = \sum_{j\in J} \sum_{i\in J_j'} u_{ij} \quad (5)$$

$$\sum_{j\in J} \sum_{i\in J_j'} u_{ij} \leq A \quad (6)$$

$$u_{kl} + \sum_{i<k\in J_l'} x_{il}^{kl} + \sum_{j\in J\setminus\{l\}} \sum_{i\in J_j'} x_{ij}^{kl} = 1, \forall l \in J, k \in J_l' \quad (7)$$

$$z_{kl} + \sum_{i>k\in J_{l,}'} x_{kl}^{il} + \sum_{j\in J\setminus\{l\}} \sum_{i\in J_j'} x_{kl}^{ij} = 1, \forall l \in J, k \in J_l' \quad (8)$$

$$c_{kl} - v_{kl} - P_{kl} \geq 0, \forall l \in J, k \in J_l' \quad (9)$$

$$c_{kl} - c_{ij} - P_{kl} \geq LN\left(w_{ij}^{kl} - 1\right), \forall j, l \in J, i \in J_j, k \in J_l (ifj = l: k > i) \quad (10)$$

$$v_{kl} - c_{(k-1)l} - \tau_{(k-1)l}^{kl} \geq 0, \forall l \in J, k \in J_l' \setminus \{1\} \quad (11)$$

$$v_{1l} - \tau_{LU}^{1l} \geq 0, \forall l \in J \quad (12)$$

$$v_{kl} - v_{ij} - \tau_{ij}^{(k-1)l} - \tau_{(k-1)l}^{ij} \geq LN\left(x_{ij}^{kl} - 1\right), \quad \forall j, l \in J, i \in J_j', k \in J_l' \setminus \{1\}, (ifj = l: k > i) \quad (13)$$

$$v_{1l} - v_{ij} - \tau_{ij}^{LU} - \tau_{LU}^{1l} \geq LN\left(x_{ij}^{1l} - 1\right), \forall j, l \in J: j \neq l, i \in J_j' \quad (14)$$

$$w_{ij}^{kl}, x_{ij}^{kl} \in \{0,1\}, \forall j, l \in J \cup \{f, t\}, i \in J_j', k \in J_l' \quad (15)$$

$$ET, c_{kl}, v_{kl} \geq 0, \forall l \in J, k \in J_l' \quad (16)$$

$$0 \leq u_{kl}, z_{kl} \leq 1, \forall l \in J, k \in J_l' \quad (17)$$

目标是最小化所有作业的退出时间，如表达式(1)中所示，其值确定为所有作业的最后(虚拟)操作的最大完成时间，如不等式(2)中所示。约束(3)和(4)要求每个操作立即紧随其后，并在同一台机器上立即进行另一个操作。约束(5)确保车辆的第一和最后任务数量相同，约束(6)确保最多为可用车辆数量。约束(7)和约束(8)分别要求每个任务必须紧随其后并紧随其后的恰好是另一个任务。由于每个车辆都通过约束(5)到(8)确定了一组链式任务，因此对车辆进行了隐式处理。因此，避免了额外的索引。 每个操作的完成时间必须满足两个约束条件。一方面，一个操作只能在作业到达处理它的机器并且它的处理时间已经过去之后才能完成，这是由约束(9)保证的;另一方面，一个操作的完成除了需要在同一台机器上完成之前的操作之外，还需要自己的处理时间，这是由约束(10)保证的。同样，一个作业只有在其上一个操作完成(c(k−1)l)并且该作业已经从 处理前一个操作的机器((τ(kl k−1)l)被约束(11)所施加的条件下，才能到达机器进行特定操作的加工;除非它是作业优先操作(k = 1)，在这种情况下，作 业可以在从 LU((τLU1l)传输后尽快到达，这是约束条件(12)所施加的。此外，如果车辆在当前作业之前立即运输了其他作业(j∈j: j = l)，则需要(i)将该作业交付给相应的机器(vij);(ii)从处理前一个操作的地方取出当前作业(τij (k−1)l)，如果它是第一个操作，则取出 LU(τij LU)，并且(iii)通过约束 (13)和(14)将其传递给相应的机器(τ(kl k−1)l 或 τLU kl)。最后，约束(15)到(17)定义了变量的性质。

2.3 下边界过程

通过一些初步实验发现，MILP 模型只能在小规模问题实例中得到最优解。对于中等规模的实例，MILP 可以给出可行解和下界。然而，两者都比 PSOSA 和下限程序分别发现的结果更差。此外，对于大型实例，它甚至无法找到任何可行的解决方案。因此，下限对于评估 PSOSA 在处理大型实例时的有效性是有用的。为此，我们提出了一个较低的边界过程，该过程考虑了一个有足够资源(机器和车辆)可用的问题版本。这样的假设意味着作业永远不会等待(被处理或运输)。 要解决的问题有三个要素，即:工作、机器和交通工具。因此，退出时间的下界必须至少大于 i)所有作业中最大的退出时间(LET J)，ii)所有机器中最大的完成时间加上相应的机器与 LU 之间的行程时间(LCT MT T)， 以及 iii)最小平均车辆行程时间

(MATT)。因此，得到了一个出口时间 LB 的下界。

$$LB = max\{LETJ, LCTMTT, MATT\} \quad (18)$$

作业(ET J J, J∈J)的退出时间定义为至少等于(i)LU 与首先处理该作业的机器之间的行程时间之和;(ii)处理剩余作业操作的机器之间的行程时间总和;(iii)加工作业最后一次操作的机器到 LU 之间的行程时间; (iv)加工作业所有操作的加工时间之和(见 Eq.(19))。所有作业中最大的退出时间(LET J)由 Eq.(20)给出。回想一下，我们假设有足够的机器和车辆，因此，工作永远不会等待。

$$ETJ_j = \tau_{LU}^{1j} + \sum_{i\in J_j\setminus\{1\}} \tau_{(i-1)j}^{ij} + \tau_{n_j j}^{IIJ} + \sum_{i\in J_j} P_{ij}, \forall j \in J \quad (19)$$

$$LETJ = \max_{j\in J}\{ETJ_j\}. \quad (20)$$

同样，一台机器(CT Mm, m∈m)的完成时间定义为它处理的所有 操作(Oij: Mij = m, i∈ J J, J∈J)中(i)最早开始时间(ESTij)与(ii)在同一台机器上处理的所有其他操作(其最早开始时间不小于操作 Oij)所需的总 处理时间之和的最大值，如式(21)所述。

$$CTM_m = \max_{j\in J, i\in J_j: M_{ij}=m}\{EST_{ij} + \sum_{l\in J} \sum_{k\in J_l} P_{kl}\}, \forall m \in M. EST_{ij} \le EST_{kl} \quad (21)$$

操作 Oij, j∈j, i∈j j 的最早开始时间 ESTij，如果是作业优先操作，则以从装载区到加工该操作的机器的行程时间为界，或者将加工该操作的机器与前一次操作的行程时间加上前一次操作的最早完成时间 ECT (i−1)j，如式(22)所示。

$$EST_{ij} = \begin{cases} \tau_{IU}^{1j}, & \forall j \in J, \\ ECT_{(i-1)j} + \tau_{(i-1)j}^{ij}, & \forall j \in J, i \in J_j \setminus \{1\}, \end{cases} \quad (22)$$

其中，操作 Oij 的最早完成时间 ECTij, j∈j, i∈j j 由其最早开始时间与处 理时间之和给出，即 ECTij = ESTij + Pij。同样，不考虑等待时间，因为我们假设有足够的资源(机器和车辆)可用。最后，所有机器中最大完成时间加上机器到 LU(LCT MT T)之间的相应行程时间由式(23)给出。

$$LCTMTT = CTM_{m^*} + \tau_{m^*}^{LU}, \text{where } m^* = \operatorname*{argmax}_{m\in M}\{CTM_m\} \quad (23)$$

通过将车辆平均装载行程与车辆平均最小所需空载行程相加，得到最小平均车辆行程时间。对于考虑 LU 并使用退出时间作为性能度 量的问题版本，可能不需要空行程，因为作业最初在 LU 可用，并且必须返回到 LU。因此，在这种情况下，车辆的最小平均行驶时间由 车辆装载的平均行驶时间给出，计算方法如式(24)所示。

$$MATT = \left\lceil \frac{1}{A}\sum_{j\in J} \left(\tau_{LJ}^{1j} + \sum_{i\in J_j\setminus\{1\}} \tau_{(i-1)j}^{ij} + \tau_{n_j j}^{IJ}\right)\right\rceil \quad (24)$$

为了得到最大跨度 Cmax 的下界，我们使用等式。(25) -(27)而不是等式。(19)、(23)和(24)，因为作业完成后不会被运送回 LU。注意，作业只在 LU 和首先处理作业的机器之间传输;因此，将会有一些空的旅行从 LU 接工作。因此，通过将平均车辆装载行程和最小平均车辆空载行程相加，得到最小平均车辆行驶时间。进一步，我们还可以在所有作业中加上最后一个作业的最小加工时间，因为只有在加工完最后一个作业之后，作业才算完成。

$$ETJ_j = \tau_{IJ}^{1j} + \sum_{i\in J_j\setminus\{1\}} \tau_{(i-1)j}^{ij} + \sum_{i\in J_j} P_{ij}, \forall j \in J \quad (25)$$

$$LCTMTT = \max_{m\in M}\{CTM_m\} \quad (26)$$

$$MATT = \left\lceil \frac{1}{A} \sum_{j \in J} \left(\tau_{LU}^{1j} + \sum_{i \in J_j \setminus \{1\}} \tau_{(i-1)j}^{ij} \right) \right\rceil + \left\lceil \frac{|J|-A}{A} \min_{m \in M} \{\tau_m^U\} \right\rceil + \min_{j \in J} \{P_{n_j j}\}. \quad (27)$$

2.4 混合算法

粒子群优化(Particle swarm optimization, PSO)是一种基于种群的随机优化技术，其灵感来自于鸟群或鱼群的社会群体行为，最初由 Eberhart & Kennedy(1995)提出。鉴于其简单的概念和有效性，PSO 已成为流行和许多变种已开发多年。在 PSO 中，一群解被称为群，每个解被称为粒子。其基本思想是在搜索空间中移动粒子，目的是找到好的解决方案。每个粒子代表 D 维搜索空间中的一个点(D 是决策的数量)，因此，它由位置向量表示。速度向量与每个粒子相关联，它定义了粒子如何在搜索空间中移动。这些向量在每次迭代时都会通过考虑三个组件进行更新，即:惯性、认知和社交组件。这些移动试图平衡向先前最佳位置移动和向最佳群体/邻居位置移动的趋势。最初，每个粒子在问题的搜索空间中是随机定位的。由于元启发式算法的性能可以通过与局部搜索过程混合来提高，因此在这项工作中，我们将 PSO 算法与 SA 算法结合起来。通过这种方式，利用了前者的探索能力和后者的局部搜索能力。局部搜索方法通常会出现过度收敛的问题，而且往往会陷入局部最优。然而，SA 能够通过偶尔接受非改进的邻居解决方案来避免这个缺点。此外，SA 不会(显著地)破坏 PSO 进化和学习过程，因为解决方案越差，被接受的可能性就越小。

2.4.1 POSA 算法框架

Algorithm 1: The proposed PSOSA algorithm for the JSPT.

Initialize parameters: S^{max}, ω, c_1, c_2, T_I, T_F, α, R;
Generate a random initial *Swarm* (position and velocity vectors for each particle):$X_s^0 = rand$, $V_s^0 = \frac{rand - X_s^0}{2}, \forall s \in S^{max}$;
Calculate objective function for each particle: $F\left(X_s^0\right), \forall s \in S^{max}$;
Initialize the personal best for each particle: $P_s = X_s^0, F(P_s) = F\left(X_s^0\right), \forall s \in S^{max}$;
Initialize the incumbent and current solutions: $a = \arg\min_{s \in S^{max}} \left\{F\left(X_s^0\right)\right\}$;
$P^* = X_a^0, F(P^*) = F(X_a^0), X^{SA} = P^*, F(X^{SA}) = F(P^*)$;
Set the initial temperature and generation counter: $T = T_I, g = 0$;
while $(T \geq T_F) \parallel (P^* > Optimal/LB)$ **do**
  for $r = 1 : R$ **do**
    Generate a neighbor solution $Y$and calculate $F(Y)$;
    Calculate $\Delta F = F(X^{SA}) - F(Y)$;
    if $\Delta F \geq 0$ **then**
      Replace the current solution: $X^{SA} = Y$, $F(X^{SA}) = F(Y)$**if** $F\left(X^{SA}\right) < F(P^*)$ **then**
        Replace the incumbent solution: $P^* = X^{SA}$, $F(P^*) = F\left(X^{SA}\right)$
      end
    else
      Calculate the acceptance probability: $Pr = e^{\frac{\Delta F}{T}}$ **if** $Pr \geq rand$ **then**
        Replace the current solution: $X^{SA} = Y$, $F\left(X^{SA}\right) = F(Y)$
      end
    end
  end
  for $s = 1 : S^{max}$ **do**
    Update the velocity of particle s: V_s^{g+1};
    Update the position of particle s: X_s^{g+1};
    Calculate $F(X_s^{g+1})$;
    if $F\left(X_s^{g+1}\right) < F(P_s)$ **then**
      Update particle's personal best: $P_s = X_s^{g+1}, F(P_s) = F\left(X_s^{g+1}\right)$
    end
  end
  $a = \arg\min_{s \in S^{max}} \{F(P_s)\}$;
  if $F(P_a) < F(P^*)$ **then**
    Update the incumbent solution: $P^* = P_a$, $F(P^*) = F(P_a)$
  end
  Update the current solution: $X^{SA} = P^*$, $F(X^{SA}) = F(P^*)$;
  Update the temperature and generation counter: $T = \alpha T, g = g + 1$,
end
Return the incumbent solution: P^* and $F(P^*)$.

图 2　POSA 算法框架

算法 1 阐述了所提出的混合粒子群优化和模拟退火算法(PSOSA)。 该算法有一个处理粒子群的外循环控制温度和两个内环。第一个内环搜索现有解决方案的邻域，第二个内环决

定是否接受更坏的邻居解决方案。该算法首先初始化参数(群体规模 Smax、惯性权重 ω、认知 c1 和社会 c2 加速系数、初始 TI 和最终 TF 温度、冷却速率 α 和内环迭代次数 R)，并生成随机群体或粒子。每个粒子(s∈Smax)初始化(在第 g = 0 代)，用两个随机向量表示它的位置(x50)和速度(Vs0)。然后构建一个可行的时间表 (见第 4.3 节)，并根据所考虑的性能指标进行评估，该指标是 ET 或 Cmax，用 F (x50)表示。将每个粒子的最佳位置和相应的度量值初始化为粒子的初始位置和度量值，即 Ps=Xs 0 和 F(Ps)= F(Xs 0)。最后，将在位粒子(P*和 F (P*))和当前解(X SA 和 F (X SA))的位置和目标函数值设置为目标函数值最优的粒子的位置和目标函数值。外环检查温度值是否保持高于最终温度:如果不是，则算法终止;否则，则降低温度。第一个内循环，重复 R 次，搜索邻居解。如果邻居解决方案比当前解决方案好，那么它替换当前解决方案，如果它也比在位解决方案好，那么它也替换现有的解决方案。然而，更坏邻居的解决方案也可以被接受作为当前解决方案的替代方案;这样就可以进行更广泛的搜索。接受的概率取决于邻居解决方案的糟糕程度，并随着解决方案空间探索的进展而降低。在第二个内环中，遵循 PSO 运动机制，粒子移动到不同的位置，得到一个新的粒子群。如果需要，将更新在位解和当前解，并重新启动外环。

2.4.2 更新粒子速度位置

通过更新粒子的位置和速度，获得了一个新的粒子群。首先更新速度，然后使用新的速度(实际上是位移)来更新位置。虽然已经提出了几个表达式来更新速度;通常，它们除了使用粒子的当前速度外，使用粒子的当前位置、粒子的最佳位置和在位者的位置。已经观察到，粒子的轨迹倾向于保持与坐标轴平行，并且所有可能的下一个位置的分布不是均匀的;在粒子当前位置周围的 d 维矩形中心附近密度更大。由于在更新速度时引入了观察到的偏差，因此 PSO-2011 标准建议以几何方式更新速度(将所有粒子定位在 d 维球体内)，见式(28)至(30)。

$$G_s^g = \frac{(x_s^g \oplus p_s^g \oplus l^g)}{3} \tag{28}$$

$$p_s^g = X_s^g \oplus c_1 U_1^g \otimes \left(P_s \ominus X_s^g\right) \tag{29}$$

$$l_s^g = X_s^g \oplus c_2 U_2^g \otimes \left(P^* \ominus X_s^g\right) \tag{30}$$

2.4.3 解表示和解码

JSPT 的解由每个有 N 个元素的实数两部分向量(粒子)表示，其中 N 是生产操作的总数，包括虚拟操作，即 N= j∈j Nj，其中 Nj= Nj + 1。元素 1 到 N 与(生产)操作相关联，按其自然顺序考虑;因此，元素 1 到 n1 是指作业 1 的操作，元素 n1 + 1 到 n1 + n2 是指作业 2 的操作，以此类推。向量第二部分的元素指的是将车辆分配给操作所需的(运输)任务，与前面一样，这些任务按其自然顺序考虑。最初，所有粒子都是按照均匀分布随机生成的;前 N 个元素在区间[0,1]内，后 N 个元素在区间[0,A] 内。解码过程有两个不同的部分，一个是获得生产操作的序列，另一个是获得车辆的分配。为了解码解向量的第一部分，我们使用最初由 Bean(1994)设计的最小位置值(SPV)规则，将实数转换为操作的排列为 了保证可行性，将操作的排列转换为对应作业的排列，然后将作业号转换为有序的操作号，即作业号第一次出现时被解码为其第一次操作，第二次出现时被解码为其第二次操作，以此类推。关于解决方案的第二部分向量,真正的 num-bers 转化为车辆数量如下:数小于或等于 1 转换为车辆数量大于 1 但小于或等于 2 翻译成 ve-hicle 2,等等,直到车辆,任意实数比(−1)翻译成汽车 A sec-ond 解决方案的一部分向量的每个元素指的是独特的任务与每个操作有关。因此，元素 N + i 为需要处理操作 i 的任务分配了一辆车。让我们借助前面的示例来说明编码和解码过程。解向量有 2 ×N = 24 个元素。该问题的初始随机粒子如图 3a 所示，使用 SPV 规则获得的 相应操

作排列如图 3b 所示。例如，由于第 5 个位置的 0.01 是最小值(在 粒子的第一部分中)，因此调度中的第一个操作是操作 5，然后是操作 7， 操作 7 是 0.05 的位置，第二个最小值，以此类推。然后将这种排列转换 成图 3c 所示的作业序列；操作 1 到 4 用 1 表示，即它们所指的作业，操作 5 到 8 用 2 表示，操作 9 到 12 用 3 表示。然后，通过用 Oij 替换作业 j 的第 i 个外观，将作业编号转换为有序操作。对于向量的第二部分，小于等于 1 的值被替换为 1，大于 1 的值被替换为 2。这样，将一个载体(元素值)分配给每个操作所需的任务(元素位置)。可行的时间表如图 3d 所示。根据得到的操作顺序和机器操作分配的输入数据(见表 2)，我们确定了每台机器的操作顺序:M1−O22, O11, O33;M2−o13, O31;M3−o12, o21;M4−O32、o23。

关于车辆的分配，车辆 1 被分配给任务 1、3、4、7、8、11，它们与操作 O11、O31、O41、O32、O42、O33。这些运算的序列，由向量的第一部分给出，是:O11, O32, O42, O31, O41, O33。因此，车辆 1 将作业 1 从 LU 传输 到机器 M11，作业 2 在机器 M22 和 M32 之间传输，然后再传输到 LU(作业 2 出口)，作业 1 在机器 M21 和 M31 之间传输，然后再传输到 LU(作业 1 出 口)，作业 3 在机器 M23 和 M33 之间传输。同样，车辆 2 将作业 2 从 LU 运 输到机器 M12，然后再运输到机器 M22，作业 3 从 LU 运输到机器 M13， 作业 1 在机器 M11 和 M21 之间运输，作业 3 在机器 M13 和 M23 之间运输， 然后再运输到 LU(作业 3 退出)。

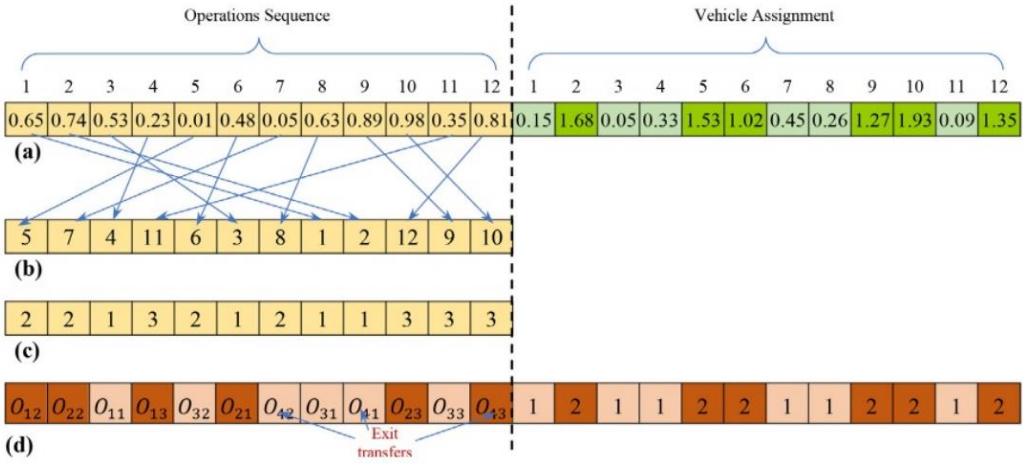

图 3 1 个初始解和相应的解码可行时间表

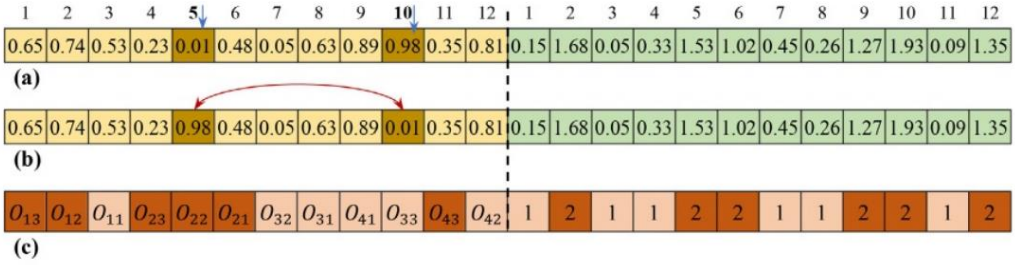

图 4 (a)原解;(b)邻居解，(c)最终可行邻居解

2.4.4 领域搜索结构

为了获得邻居解，我们首先生成一个介于 0 和 1 之间的随机数:如果 它小于 0.5，那么我们交换解向量第一部分的两个元素;否则，我们交换第二部分的两个元素。在这两种情况

下，我们都会生成两个随机数(在 1 和 N 之间或在 N + 1 和 2N 之间，这取决于我们是交换解向量第一部分的元素还是第二部分的元素)来确定要交换的元素。所得到的邻居解可能对机器和车辆有不同的操作顺序，或者对车辆有不同的赋值，这取决于被交换的元素是在向量的第一部分还是第二部分。图 4 通过生成图 3a 所描述的邻居解来说明该过程的应用，为了方便起见，在图 4a 中重复了这个过程。假设要在解向量的第一部分上执行交换，并且生成的两个随机数为 5 和 10。编码后的邻居解如图 4b 所示，对应的可行调度如图 4c 所示。

3. 论文算法求解

论文在三组基准问题实例 2 上对所提出的 MILP 模型、LB 方法和 PSOSA 的有效性和效率进行了评估，即 Bilge & Ulusoy(1995)提出的 BU 实例、Hurink & Knust(2005)提出的 HK 实例和 Ham(2020)提出的 SWV 实例。所获得的结果与当前最先进的方法进行了比较。

3.1 BU 实例

PSOSA 为所有实例找到了最优/最知名的解决方案，除了实例 EX71;优于 MA 和 ALS，后者分别有 2 次和 3 次未能做到这一点。对于没有已知最优解的两个实例，PSOSA 能够比已知解平均提高 0.32%，而 MA 和 ALS 分别提高 0.29%和 0.24%。此外，PSOSA 的稳健性可以从退出时间百分比标准差的低值得到，其范围为 0～2.53%，平均值为 1.23%。

关于计算时间，在某些情况下，MILP 很快就得到了最优解，在这些情况下，PSOSA 需要几秒钟的额外时间。然而，PSOSA 的平均计算时间比 MILP 要小得多。平均而言，PSOSA 需要 9.6 秒(0.2 到 18.9 秒)，而 ALS 需要 1117 秒。9 秒(范围从 3.5 到 6996.6 秒)。在计算次数方面，与优化退出时间时的计算结果相似。尽管 PSOSA 在两个实例中的计算时间比 MILP 要长，但对于其他实例来说，PSOSA 的计算时间要快得多。PSOSA 和 MILP 的平均计算时间分别为 3.8 秒和 312.6 秒。然而，PSOSA 对于所有实例都要快得多。平均需要 3.8 秒，而 ALS 需要 559 秒。

3.2 HK 实例

结果显示，PSOSA 明显优于其他三种算法，因为除了两个实例外，它总是找到一个同样好或更好的解决方案。此外，最大完工时间的平均百分比标准差非常小(1.53%)。对于个体比较，PSOSA 匹配了 6 个实例的最佳 MA 解，对 16 个实例的最佳 MA 解进行了改进，并对剩下的一个实例找到了更差的解；PSOSA 平均缺口比 MA 低 3%。PSOSA 的计算时间范围为 11.02~159.47 秒，所有实例的平均值为 93.43 秒，P1 和 P2 实例的平均值分别为 28.2 秒和 122 秒。

3.3 SWV 实例

从结果可以看出，PSOSA 优于 CP2，因为在考虑的 10 个实例中，它分别为 3 辆车和 5 辆车的车队规模找到了 9 个和 7 个更好的解决方案。此外，PSOSA 的平均差距低于 CP2，三辆车和五辆车的车队分别低约 3%和 1%。本文首次报告了考虑退出时间的 SWV 实例的结果。在存在较大的问题实例时，PSOSA 仍然是健壮的。三种情况下最大完工时间和退出时间的平均百分比标准差分别为 1.18%和 0. 98%，当考虑 5 辆车时，分别为 1.55 和 1.49%。

4 论文复现

4 1 事例复现

我们首先对论文中给出的解释案例的事例进行了实验，得出的甘特图和适应度曲线见图 5 及图 6。

Table 2
Illustrative example: instance data.

Jobs	Job 1			Job 2			Job 3		
Operations	O_{11}	O_{21}	O_{31}	O_{12}	O_{22}	O_{32}	O_{13}	O_{23}	O_{33}
Machine	M1	M3	M2	M3	M1	M4	M2	M4	M1
Processing time	10	16	18	15	22	18	18	16	22

Table 3
Illustrative example: transport times, layout 1 in (Bilge & Ulusoy, 1995).

	LU	M1	M2	M3	M4
LU	0	6	8	10	12
M1	12	0	6	8	10
M2	10	6	0	6	8
M3	8	8	6	0	6
M4	6	10	8	6	0

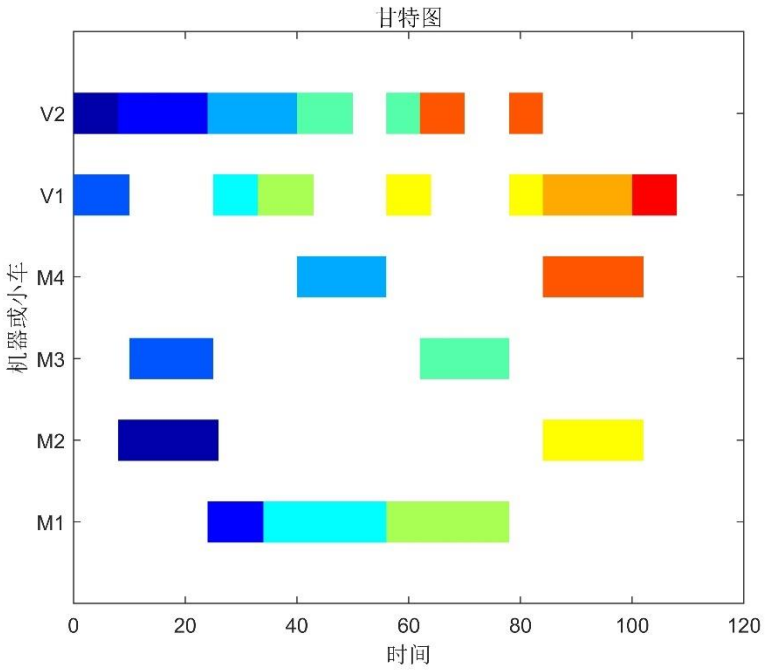

图 5 甘特图

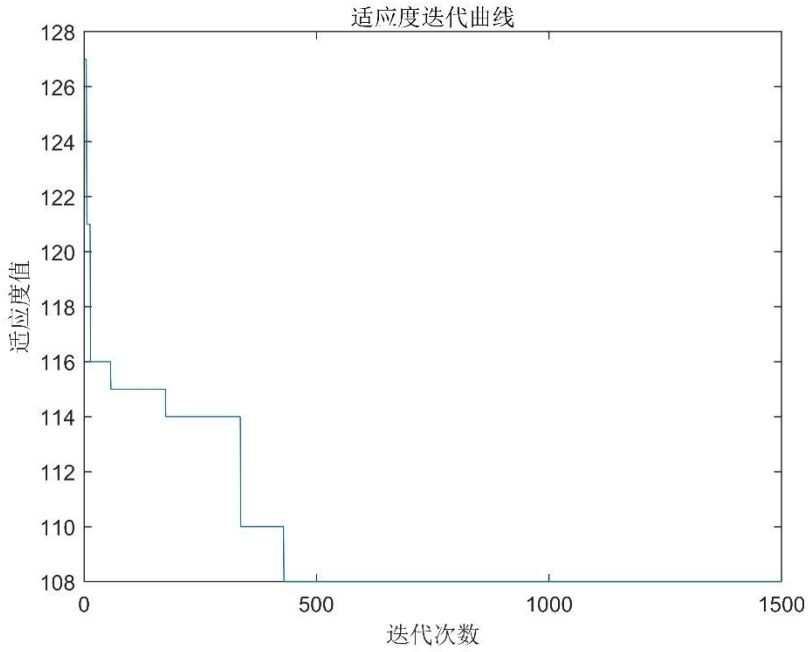

图 6 适应度迭代曲线

4.2 算法复现

采用 MATLAB 对 PSOSA 算法进行复现，由于时间问题，我们只对 BU 数据集进行了测试，每个实例各运行了两次，运行结果见附录。

观察运行结果，总计四十个实例中，第一次复现有 4 个实例的 ET 值优于原论文结果，7 个实例的 ET 值和原论文相同，其余实例均未达到最优解，第二次复现有 7 个实例的 ET 值优于原论文结果，1 个实例的 ET 值和原论文相同，其余实例均未达到最优解。该算法一定程度上优化了先前的算法，但仍然存在着精确度不够、稳定性不足等问题，仍需改进。

注：1、本次复现代码见压缩文件 PSOSA.zip。

2、本课程所有代码见 Github 仓库

https://github.com/Erpeng-Zhang/algorithm-CWCclass.git

附录 1 复现结果

Name	原论文结果			复现第一次结果			复现第二次结果		
	ET	GAP	T	ET	GAP	T	ET	GAP	T
EX11	114	0	2.9	112	-0.02	6.4	113	-0.01	8.5
EX12	90	0	2.9	90	0.00	3.6	92	0.02	250.7
EX13	98	0	1.9	98	0.00	1.8	102	0.04	312.7
EX14	140	0	2.9	140	0.00	0.2	134	-0.04	0.3
EX21	116	0	3.5	124	0.07	291.8	130	0.12	345.1
EX22	82	0	3.1	104	0.27	291.1	110	0.34	295.2
EX23	89	0	3.6	114	0.28	290.6	115	0.29	551.9
EX24	134	0	11.7	154	0.15	290.1	156	0.16	523.8
EX31	121	0	13.4	136	0.12	291.5	141	0.17	1361.9
EX32	89	0	5.4	110	0.24	289.6	115	0.29	235.5
EX33	96	0	4.6	120	0.25	291.8	124	0.29	191.1
EX34	148	0	13.2	154	0.04	291.5	155	0.05	191.3
EX41	136	0	14.1	136	0.00	51.7	132	-0.03	55.7
EX42	100	0	5.1	120	0.20	261.1	117	0.17	172.0
EX43	102	0	14.9	116	0.14	260.8	115	0.13	172.1
EX44	163	0	13.2	162	-0.01	48.8	162	-0.01	28.3
EX51	110	0	3.7	110	0.00	0.3	108	-0.02	3.9
EX52	81	0	4.1	80	-0.01	66.3	81	0.00	35.3
EX53	89	0	4.5	89	0.00	17.6	88	-0.01	0.2
EX54	134	0	4.8	127	-0.05	0.1	125	-0.07	0.1
EX61	129	0	14.9	147	0.14	290.7	145	0.12	193.0
EX62	102	0	9.2	128	0.25	291.2	132	0.29	192.9
EX63	105	0	15.3	138	0.31	292.1	143	0.36	192.8
EX64	151	0	15.8	176	0.17	291.9	171	0.13	192.6
EX71	134	-8.22	18	146	0.00	18.7	147	0.01	242.2
EX72	86	0	17.1	125	0.45	360.6	117	0.36	241.9
EX73	93	0	18.9	132	0.42	361.9	125	0.34	239.7
EX74	161	-4.73	17.5	176	0.04	362.1	189	0.12	239.9
EX81	167	0	1.9	184	0.10	295.4	187	0.12	195.0
EX82	155	0	0.2	165	0.06	292.3	164	0.06	195.1
EX83	155	0	0.3	179	0.15	292.6	189	0.22	195.0
EX84	178	0	15.4	212	0.19	292.6	214	0.20	195.4
EX91	127	0	11	147	0.16	255.8	147	0.16	170.5
EX92	106	0	5.8	131	0.24	259.3	126	0.19	170.5
EX93	107	0	11.1	134	0.25	259.2	135	0.26	170.5
EX94	149	0	12.5	161	0.08	256.9	160	0.07	169.6
EX101	153	0	16.2	186	0.22	293.7	182	0.19	302.4
EX102	139	0	15.6	176	0.27	294.5	161	0.16	239.6
EX103	139	0	18.8	177	0.27	293.6	191	0.37	194.9
EX104	183	0	15.3	218	0.19	294.9	196	0.07	195.4