使用遗传算法优化生产线成本

Alireza Rezaee机电一体化系，智能系统学院，跨学科科学与技术学院，德黑兰大学，伊朗德黑兰
arrezaee@ut.ac.ir

arXiv:2601.00689v1 [cs.NE] 2 Jan 2026

2026年1月5日

摘要

本文提出了一种采用遗传算法（GA）实现生产线成本最优任务调度的方法。该系统由一组串行加工任务组成，每项任务均具有给定的持续时间、单位执行成本和优先约束；这些任务必须在满足每个工位时长上限的条件下，分配至数量不限的工位。目标是在严格满足所有前置任务约束和容量约束的同时，最小化总生产成本；该成本建模为任务成本与时长上限的工位级函数。本文研究了两种染色体编码策略：一种是基于工位的表示，使用 JGAP 库并通过 SuperGene 进行有效性检查；另一种是基于任务的表示，其中基因直接编码工位分配。针对每种编码方式，均对标准遗传算法算子（交叉、变异、选择和替换）进行了调整，以保持可行性并推动种群趋向成本更低的调度方案。在三类优先结构——紧密耦合、松散耦合和非耦合——上的实验结果表明，与基于工位的编码相比，基于任务的编码能够实现更平稳的收敛和更可靠的成本最小化，尤其是在可行调度方案数量较多时。研究凸显了遗传算法相较于基于梯度的方法和解析方法在组合调度问题中的优势，特别是在存在复杂约束和不可微成本曲面的情况下。

1 遗传算法简介

自然界有一套行之有效的机制，能够演化出适应力强的生物。不适应环境的生物会逐渐消亡，而适应环境的生物则能存活下来并繁衍后代。后代与父代相似，因此每一代的新生物都与上一代中适应力强的个体相似。如果环境缓慢变化，物种便能逐渐随之演化；但环境骤变很可能会使一个物种灭绝。偶尔会发生随机变异，尽管其中大多数会导致变异个体很快死亡，但有些变异却会催生新的成功物种。达尔文的著作《物种起源》（全名《论自然选择》）的出版，是科学史上的一个重大转折点。

事实证明，对自然有益的事物，对人工系统也有益 [5–42]。遗传算法的伪代码以一个或多个个体组成的集合为起点，并通过选择和繁殖算子使个体“进化”，从而产生由适应度函数衡量的成功个体。个体的选择方式有多种。个体可以是完整的智能体函数，此时适应度函数是性能度量或奖励函数，与自然选择的类比最为贴切。个体也可以是智能体的组件函数，此时适应度函数就是评价函数。或者，个体也可以是任何能够表述为优化问题的对象。

由于进化过程根据选择函数提供的偶尔奖励（后代）来学习智能体函数，因此可以将其视为一种强化学习。遗传算法直接在个体空间中搜索，目标是找到一个使适应度函数最大化的个体。它采用并行搜索，因为种群中的每个个体都可以视为一次独立搜索。它也是一种爬山法，因为我们对个体进行微小的遗传变化，并选用由此产生的最佳后代。关键问题是如何分配搜索资源：显然，我们应该把大部分时间花在最有希望的个体上，但如果忽略得分较低的个体，就有可能陷入局部最大值。可以证明，在某些假设条件下，遗传算法能够以最优方式分配资源。

通常，大多数遗传算法中只有两个主要组成部分取决于具体问题：问题编码和评价函数。

任何遗传算法实现的第一步都是生成一个初始种群。如图1所示，一代分为选择阶段和重组阶段。该图展示了在选择过程中，字符串被分配到相邻的槽位中。实际上，为了打乱中间种群，也可以将它们随机分配到各个槽位。交叉后可以进行变异（图中未显示）。

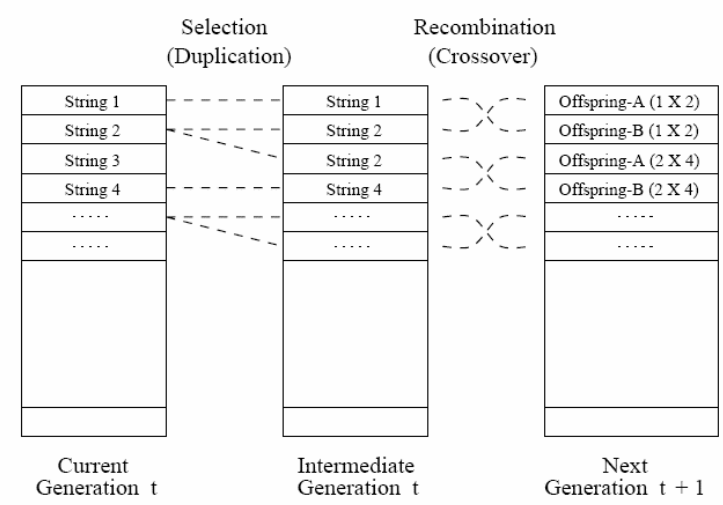


图1：进化过程（示意图）。

2 问题定义

一条生产线由 N 加工任务 T_i （$i = 1..N$）组成，这些任务应按顺序执行以生产产品。每项任务 T_i 都有持续时间 D（T_i）和单位成本 C（T_i），分别表示任务的执行时长（小时）和每小时的执行成本。因此， T_i 的总执行成本可计算为 D（T_i） $\times C$（T_i）。此外，有些任务是其他任务的前置任务；我们用 Pre(T_i, T_j) 表示，这意味着 T_i 是 T_j 的前置任务，应先于它完成。显然，前置关系具有传递性，也就是说，Pre(A, B) 和 Pre(B, C) 意味着 Pre(A, C)。

另一方面，每项任务都应在生产线上的某个工位完成；然而，一个工位可以依次处理多项任务。假设存在无限多个按串联方式排列的工位。此外，每个工位都有一个时长上限 K，规定分配给该工位的各项任务的持续时间之和不得大于 K。

Let

$$S_j = \{T_i \mid T_i \text{ is assigned to station } S_j\}$$
$$\text{Cost}(S_j) = K \times \max_{T_i \in S_j} C(T_i)$$
$$\text{Total Cost} = \sum_j \text{Cost}(S_j)$$

该问题旨在将 N 加工任务分配到各工位，使总开销最小，同时满足前置任务关系和持续时间上限约束。

3 问题编码的两种方法

本节提出了两种问题编码方案。第一种称为 基于工位 ，第二种称为 基于任务。这两种方案的区别在于它们为该问题设计的染色体结构不同。基于工位的方法使用 JGAP 库实现，而第二种方法则完全自行实现。

4 基因组表示

4.1 基于工位

使用的 JGAP 类

Gene

public interface 基因 extends java.lang.Comparable,
java.io.Serializable

基因表示潜在解（染色体）的离散组成部分。此接口旨在让自定义基因实现能够轻松接入，从而为许多应用带来很大的灵活性和便利性。请注意，此接口的实现也必须实现 equals() 方法，这一点非常重要。如果没有正确实现 equals()，某些遗传操作将无法正常工作。

void setAllele(java.lang.Object a_新值)

设置此 基因的值为给定的新值。该值的实际类型取决于具体实现 t.

IntegerGene

public class IntegerGene extends NumberGene implements Gene

一种支持其等位基因为整数值的 Gene 实现。可以选择性地提供上界和下界，以限制此 Gene 实例允许使用的合法值范围。

SuperGene

public interface Supergene extends Gene

超基因代表多个基因，这些基因通常控制表型中密切相关的方面。超基因仅以能够保持等位基因组合有效的方式发生变异。会使等位基因组合失效的变异会在 Gene.applyMutation(int, double) 方法中被拒绝。超基因组件也可以是超基因，从而形成树状结构。

在生物学中，无效组合代表完全损坏的代谢链、失衡的信号通路（存在激活因子而没有抑制因子）等情况。

随机生成的超基因超等位基因值中，至少约有 5% 应当是有效的。如果有效组合在所有可能组合中所占的比例过小，则可能需要很长时间才能找到不会破坏超基因的合适变异。如果遇到此问题，请尝试将超基因拆分为多个子超基因。

为了使染色体适用于该问题，规定每个基因只包含一项任务。所有染色体均由固定的基因集合构成，彼此之间唯一的差异在于基因的排列顺序。
由于每条染色体中的基因之间存在先决关系，因此最好使用 SuperGene，即一个能够自动检查新染色体中所需关系是否正确的基因池。
定义一个名为 SuperGene 的类，使其继承自 abstractSupergene（一个 JGAP 库类），并实现 JGAP 类的 IsValid() 函数，以检查染色体在交叉和/或变异后的有效性。
为了使用 JGAP 标准染色体，必须向 setSampleChromosome() 函数传入一个有效的示例染色体。染色体是由基因组成的数组；因此，为了使用 SuperGene，需要将染色体定义为长度为 1 的临时基因数组，并将该数组唯一的索引位置填入已定义的 SuperGene 实例。通过这一技巧，可以得到一个由长度为 1 的基因数组组成的染色体，其唯一的块是一个由 IntegerGene 填充的 SuperGene。

public class SuperGene extends abstractSupergene { ... };

```
SuperGene superGene = new SuperGene(geneArray);
基因[] temp1 = new Gene[1]; temp1[0] = superGene;
```

染色体

public class Chromosome extends java.lang.Object
implements java.lang.Comparable,
java.lang.Cloneable, java.io.Serializable

染色体表示潜在解，由固定长度的基因集合组成。每个基因代表解的一个离散部分。染色体中的每个基因都可以由 Gene 接口的不同具体实现来支持，但同一种群（基因型）内各条染色体在对应位置（基因位点）上的基因必须采用相同的具体实现。换言之，一条染色体中的基因 1 必须与种群中所有其他染色体中的基因 1 采用相同的具体实现。
染色体的实现如下：

```
基因[] 样本基因 = new Gene[Main.numberOfTasks];

for (int i = 0; i < Main.numberOfTasks; i++) sampleGenes[i] =new
IntegerGene(0, Main.numberOfTasks - 1);

SuperGene superGene =new SuperGene(sampleGenes); 基
因[] temp1 = new Gene[1]; temp1[0] = superGene;
config.setSample染色体(new Chromosome(temp1));
```

JGAP 的交叉和变异函数没有被重写。这些操作完全交由 JGAP 定义的函数处理。唯一需要处理的流程是在交叉或变异后检查所得染色体的有效性。这是通过重写 JGAP 的 abstractSupergene 类的 IsValid() 函数来完成的。

重写后的 IsValid() 函数由三个检查部分组成：

- 检查染色体中每个基因是否存在；如果不存在，则该染色体无效。
- 检查染色体中是否存在重复任务；如果存在，则该染色体无效。
- 检查当前任务的前置任务是否排在它之后；如果是，则该染色体无效。

无效染色体将从染色体池中移除。

4.2 基于任务

在此方案中，基因只是一个整数值。染色体的长度等于任务数量。每条染色体中元素基因[i] 的值表示处理任务编号 i 的工位编号。这意味着索引是通过任务编号进行的。

存在一个函数，用于根据问题的约束检查染色体是否有效。第一项约束是，任何工位所包含的一组任务，其持续时间总和不得大于 K。检查方法很简单：扫描一次染色体，并计算各工位的持续时间总和，即如果基因[i] = j，则将任务编号 i 的持续时间加到工位 j 的总持续时间中。如果任何时候工位持续时间超出上限K，该染色体就无效。
第二个约束是前置条件满足。只要分配给 T_i 的工位编号小于或等于分配给 T_j 的工位编号，前置条件 Pre(T_i, T_j) 就满足，即基因[i] <= 不大于基因[j]。由于每项任务都有一个作为输入提供的前置任务列表，只需扫描该列表一次，就可以检查上述条件。因此，检查染色体有效性的总开销为 $O(n^2)$。

5 交叉算子

5.1 基于工位

public class CrossoverOperator extends java.lang.Object
implements 遗传算子

交叉算子从种群中随机选取两条染色体，并通过随机选取一个基因，然后交换两条染色体中从该基因开始及其之后的所有基因来使它们“配对”。随后，将修改后的两条染色体添加到候选染色体列表中。此 CrossoverOperator 支持固定和动态交叉率。固定交叉率是指用户在构造时指定的交叉率。该操作执行 $1/m_crossoverRate$ 次，即种群中的染色体数量。若未提供固定交叉率，则由此类确定动态交叉率。

5.2 基于任务

交叉操作由一个独立的类实现。其目标是根据给定的父代生成两个有效子代。该函数首先随机选择一个交叉点。然后，将第一个父代的第一个槽位与第二个父代的第二个槽位拼接，生成第一个子代。第二个子代采用对称的操作。由于生成有效染色体的成本过高，且实现所有错误检查容易出错，因此该过程会先生成一个子代，再检查其有效性。无效子代会被丢弃，并继续执行该过程，直到生成两个有效子代。因此，该算法的阶为 $O(n \times x)$，其中 n 是任务数量， x 是一个非确定性值，表示需要更改交叉点的次数，以生成有效子代。

6 变异算子

6.1 基于工位

public class MutationOperator extends java.lang.Object
implements 遗传算子

变异算子会遍历种群中每个染色体的基因，并根据给定的变异率按统计概率使其发生突变。突变后的染色体随后会被添加到候选染色体列表中，供自然选择过程使用。

此 MutationOperator 支持固定和动态变异率。固定变异率由用户在构造时指定。如果未提供固定变异率，则由此类确定动态变异率，并根据种群中染色体的大小计算，使得平均而言，该算子每处理十个染色体就会突变一个基因。

6.2 基于任务

上一节中产生的每个子代都会以固定概率进行变异。在此方法中，会随机选择染色体基因数组中的两个索引。由于变异只是对染色体进行一种非常规的改变，因此该函数也会改变这两个选定元素的值。在这个问题中，此过程意味着选择两个任务，并交换分配给它们的工位。由于交换操作可能会产生无效染色体，因此程序会在最后检查其有效性，并且只有在完成有效变异后函数才会返回。值得一提的是，并非所有子代都会发生变异。变异次数为 $O(x)$，其中 x 是一个非确定值，表示产生一次有效变异所需的次数。

7 适应度函数

7.1 基于工位

通过尽可能多地将任务分配到一个工位来计算适应度值。随后，计算分配给每个工位的任务的最高成本，并乘以常数 K。该结果的倒数即为染色体的适应度值。

7.2 基于任务

通过扫描一次染色体，计算分配给每个工位的任务的最高成本。然后将这些最高成本相加，再乘以值 K。由此得到所有工位的总成本。适应度值将为总成本的 $1/$，使各代逐步趋向于产生最低成本。

8 选择与替换

8.1 基于工位

JGAP 库支持一个类，用于在每次进化过程中选择要进行交叉和变异的染色体。名为 WeightedRoulette 的类实现了一种选择算法，它会以适应度$[i]$/总适应度的概率，简单地选择编号为 i 的染色体。在配置设置的开头将此选择器添加到配置中，以便在每次进化开始时将此方法应用于种群。

程序配置为在若干次进化过程中保持种群规模不变。此外，还有一项设置可用于在世代更替过程中保留适应度最高的个体。通过这些设置，每次进化结束时，都会选出适应度最高的个体进入下一代，其余名额则通过加权轮盘赌选择下一代种群。

8.2 基于任务

根据个体的适应度来选择参与交叉操作的个体。首先，计算染色体中每个基因的 fitness$[i]$/total fitness。将这个分数命名为 proportion$[i]$。接下来，按以下方式将这些分数相加：

$$\text{proportion}[i] = \left(\sum_{j=1}^{i-1} \text{proportion}[j]\right) + \text{proportion}[i]$$

显然，上述表达式会产生总和为 1 的浮点数，即 proportion$[n] = 1$。接下来，算法使用均匀随机数生成器生成一个介于 0 和 1 之间的浮点数。如果该数落在 (proportion$[i]$, proportion$[i+1]$) 范围内，则选择值 $i+1$ 参与交叉。

值得一提的是，交叉算子被调用的次数取决于一个名为 numberOfCandidateParents 的常数字段。经过交叉和变异操作后，种群规模将增长至2 ×numberOfCandidateParents +initialPopulationSize。因此，为了保持种群规模不变并限制个体的产生，必须删除部分个体。在此阶段，算法使用随机数生成器生成一个随机整数值，该值是要从种群中移除的染色体的索引。该染色体可能是原有的父代，也可能是新生成的子代。

9 结果

结果通常针对基于任务的解决方案给出。由于第一种方法的适应度评估并不总能产生最佳世代。尽管基于工位的解决方案中的适应度算法试图将尽可能多的任务放入一个工位，但这可能无法产生适当的结果。因为在这个问题中，尽量减少所使用的工位数量并不是需要考虑的问题。因此，第一种方法无法产生所需的世代。但是，由于基于任务的方法会为每个任务明确分配工位，因此不会出现上述错误。
结果分为以下三个类别进行分析。

9.1 情况 I：紧密耦合

在这种情况下，只允许以串行方式执行单一任务序列。由于约束过于严格，遗传算法将在早期代（100 代之前）收敛，如平均适应度图所示。先决关系如图 2 所示。

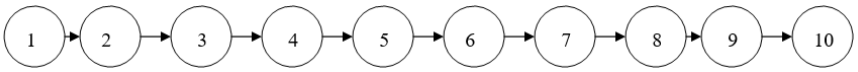


图 2：情况 I，先决关系图。

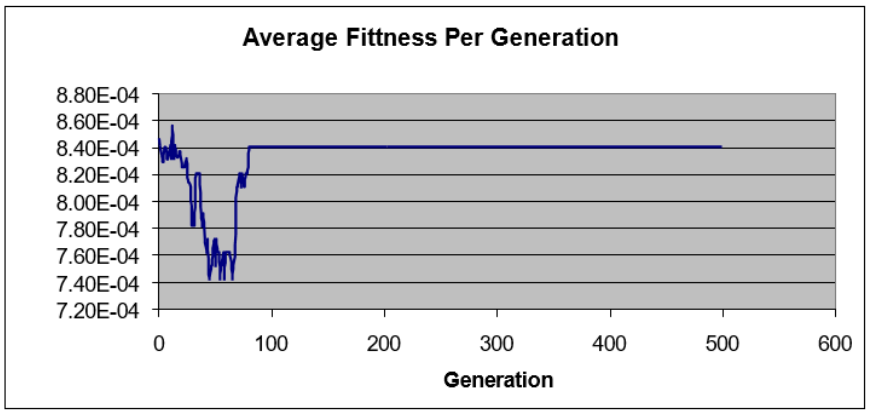


图3：情况 I，每代的平均适应度。

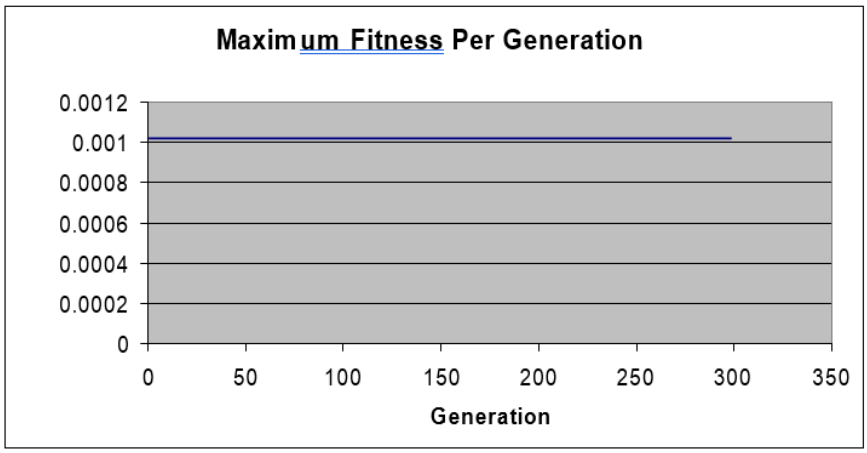


图4：情况 I，每代的最大适应度。

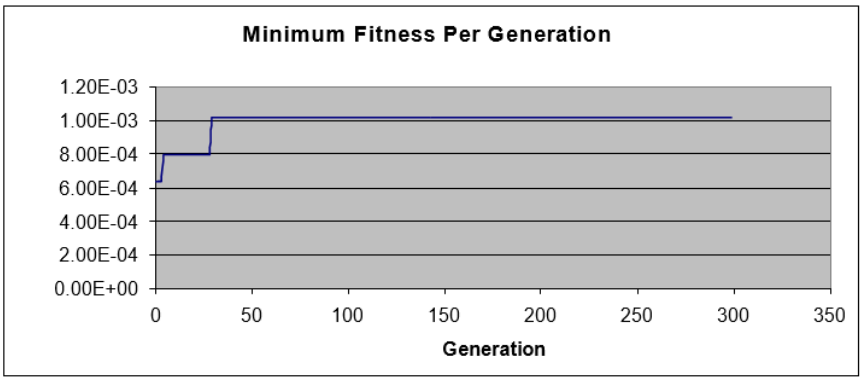


图5：情况 I，每代的最小适应度。

9.2 情况II：松散耦合

在这一类中，任务之间存在一些先决关系，但这些关系有所放宽，因此可能的有效任务序列数量相当可观。如图7所示，平均适应度有轻微波动，但使用移动窗口可以看出，它是一个递增函数，这在逻辑上是必然的。最小适应度和最大适应度图不像情况 I 中那样平坦。

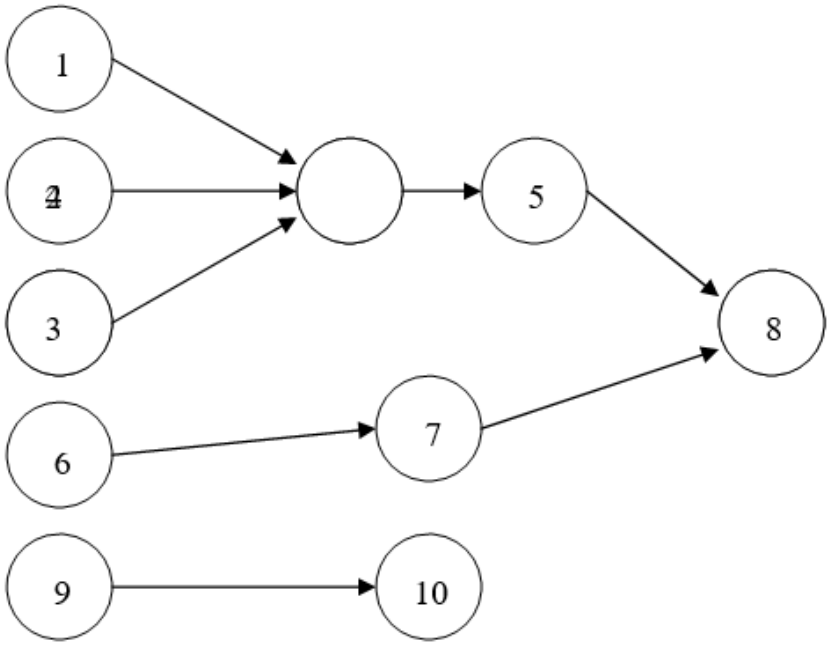


图6：情况II，先决关系图。

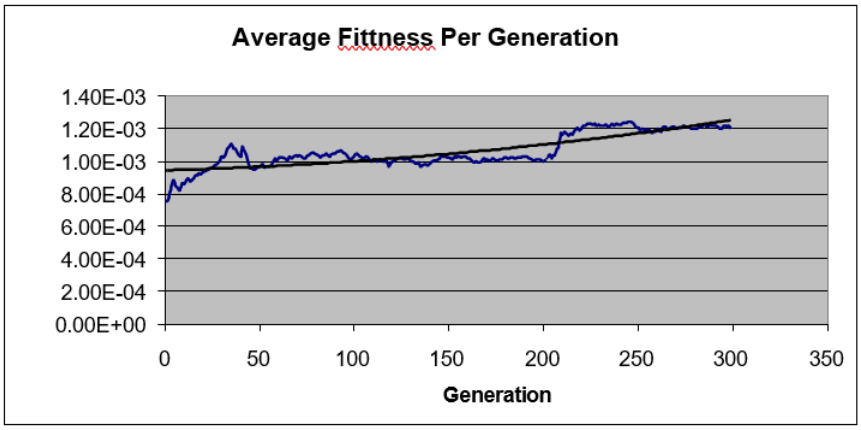


图7：情况II，每代平均适应度。

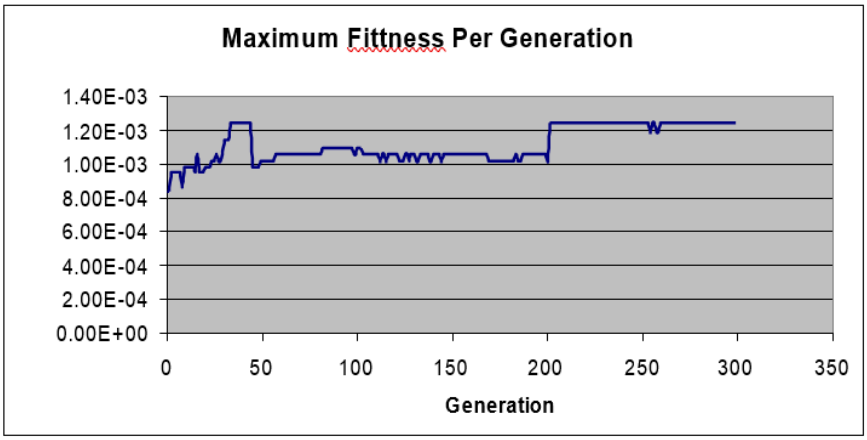


图8：情况II，每代最大适应度。

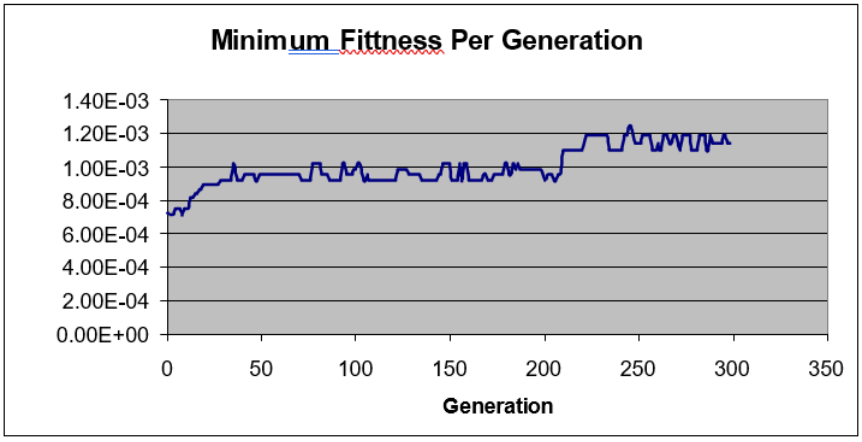


图9：情况II，每代最小适应度。

9.3 案例 III：无耦合

此类包含没有先决关系的测试用例。因此，程序运行期间唯一的检查项是最大总工位时长。对于这个特定的问题输入，也给出了第一种方法——基于工位的方法——得出的结果。不过，从图 10 和图 11 可以看出，基于任务的方法所得的图表更加平滑，因此能够更早收敛。

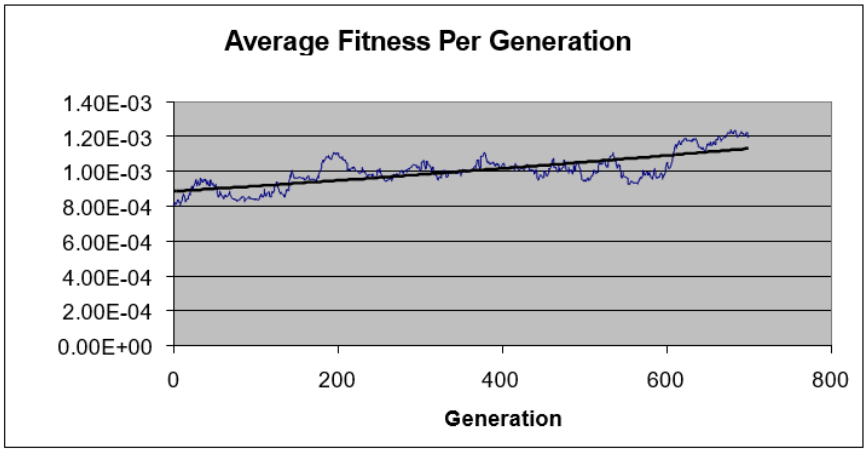


图10：案例 III，平均适应度，基于任务。

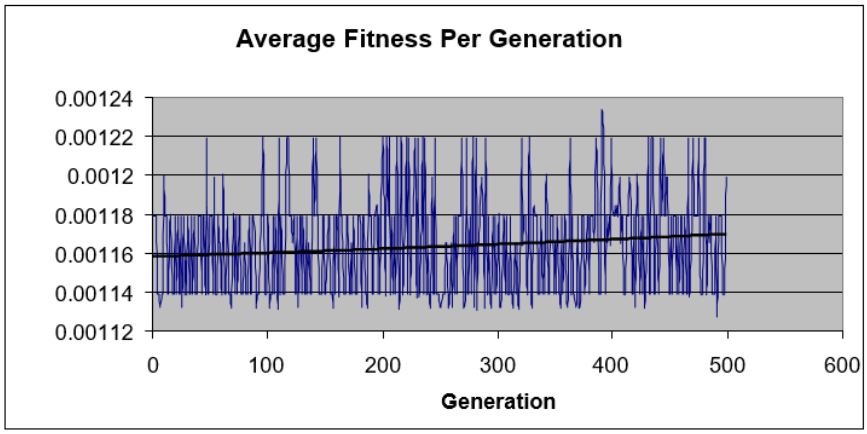


图11：案例 III，平均适应度，基于工位。

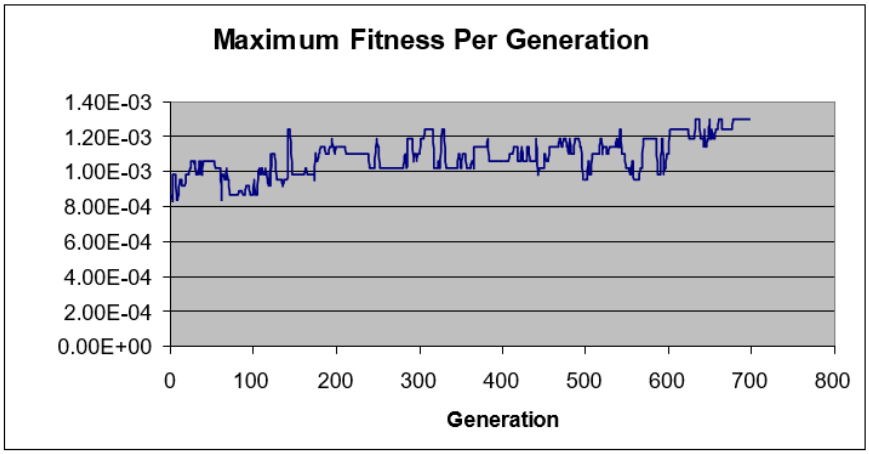


图12：案例 III，最大适应度。

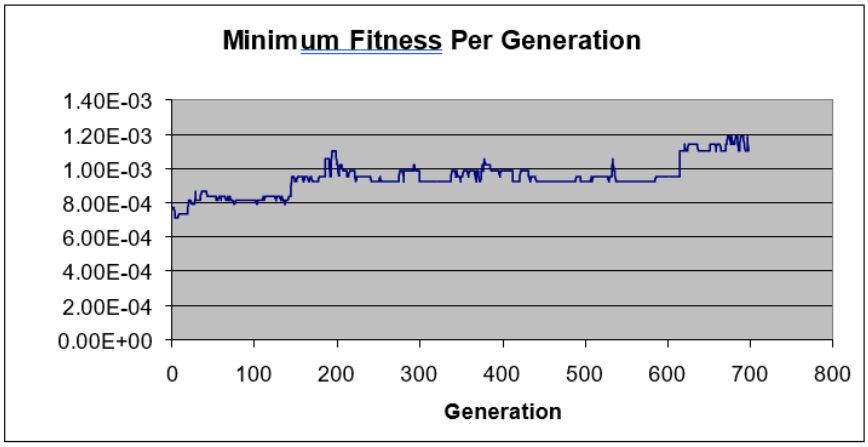


图13：案例 III，最小适应度。

10 优化技术

10.1 解析法

给定 $y = f(x)$，求 f 关于 x的导数，将结果设为零，并求解 x。这种方法效果很好，但仅适用于简单的解析函数。

10.2 基于梯度的方法（爬山法）

给定 $y = f(x)$，选取一个点 x_0，计算 $f(x_0)$ 的梯度，沿梯度方向移动以得到 $x_1 = x_0 + \alpha f(x_0)$，并重复此过程，直至得到极值。这种方法要求导数存在，而且容易陷入局部极值。

10.3 枚举法

按顺序测试状态空间中的每个点。

10.4 随机法

按顺序测试状态空间中的每个点。

10.5 遗传算法

这种方法不需要导数，只需要一个评价函数（适应度函数）。它像枚举法或随机算法一样广泛地对空间进行采样，但效率更高。这种解决方案可以并行搜索多个峰值，因此相比基于梯度的方法，受局部极值的影响更小。交叉能够组合有用的构建模块或模式（变异可以避免进化死路），而且最终具有很强的鲁棒性！

11 结论

通过以上讨论可知，基于梯度的技术可能陷入局部最大值，无法遍历整个状态空间。遗传算法不会出现这种问题。此外，也存在一些能够避免基于梯度的方法陷入局部极值的启发式方法。对于难以推导的问题，爬山法也并非首选。在这个特定问题中，采用基于梯度的解决方案存在陷入局部最大值的风险，最终可能得不到最优的任务调度。只要采用合适的启发式方法来避免陷入局部最大值，就可以将所有染色体（所有基因排列）作为 x轴，并将个体适应度作为y 轴，从而将爬山法应用于该问题。

参考文献

[1] 遗传算法简介。http://cs.felk.cvut.cz/~xobitko/ga/[2] 遗传算法。http://www.ai-junkie.com/ga/intro/gat1.html[3] 关于JGAP。http://jgap.sourceforge.net[4] 关于JGAP。http://www.jgap.org[5] S. Abdoli和F. Hajati，使用测地导数模式进行离线签名验证，载于2014年第22届伊朗电气工程会议（ICEE），2014：IEEE，第1018–1023页。[6] F. Ayatollahi、A. A. Raie和F. Hajati，使用深度与强度双树复小波变换特征进行表情不变人脸识别，《Journal of Electronic Imaging》，第24卷，第2期，第023031-023031页，2015年。[7] L. Barolli，《Advanced Information Networking a nd Applications》：第38届高级信息网络与应用国际会议（AINA-2024）论文集，第2卷，编者：Springer Nature，2024年。[8] L. Barolli、P. Hellinckx和T. Enokido，《Advances on Broad-Band Wireless Computing, Communication and Applications》：第14届宽带无线计算、通信与应用国际会议（BWCCA-2019）论文集。Springer Nature，2019年。[9] L. Barolli、M. Takizawa、F. Xhafa和T. Enokido，《Web、人工智能与网络应用》：第33届高级信息网络与应用国际会议（WAINA-2019）研讨会论文集。Springer，2019年。[10] R. Barzamini、F. Hajati、S. Gheisari和M. Motamadinejad，使用多层感知器和模糊推理系统进行短期负荷预测，《Journal of Applied Sciences》，第12卷，第1期，第40-47页，2012年。[11] D. Cremers、I. Reid、H. Saito和M.-H. Yang，《Computer Vision–ACCV 2014》：第12届亚洲计算机视觉会议，新加坡，新加坡，2014年11月1日至5日，修订版精选论文，第V部分。Springer，2015年。[12] S. Fiorini、F. Hajati、A. Barla和F. Girosi，通过时间跨度引导神经注意力网络预测澳大利亚人口中的糖尿病二线治疗启动，《PLOS ONE》，第10卷，第14期，第e0211844页，2019年。[13] F. Hajati、A. Cheraghian、S. Gheisari、Y. Gao和A. S. Mian，用于三维可变形纹理匹配的表面测地模式，《Pattern Recognition》，第62卷，第21-32页，2017年。[14] F. Hajati、K. Faez和S. K. Pakazad，彩色图像中的一种高效人脸定位与识别方法，载于《Systems, Man and Cybernetics》，2006年SMC'06 IEEE国际会议，2006年，第5卷：IEEE，第4214-4219页。[15] F. Hajati、A. A. Raie和Y. Gao，使用测地纹理变形进行姿态不变2.5D人脸识别，载于2010年第11届控制、自动化、机器人与视觉国际会议，2010年：IEEE，第1837-1841页。[16] F. Hajati、M. Tavakolian、S. Gheisari、Y. Gao和A. S. Mian，使用导数稀疏表示进行动态纹理比较：应用于基于视频的人脸识别，《IEEE Transactions on Human-Machine Systems》，第47卷，第6期，第970-982页，2017年。

[17] P. Mahajan、S. Uddin、F. Hajati、M. A. Moni和E. Gide，使用多个数据集对用于疾病预测的机器学习集成方法进行比较评估，《Health and Technology》，第14卷，第3期，第597-613页，2024年。[18] S. K. Pakazad、K. Faez和F. Hajati，基于人脸组件中心几何矩的人脸检测，载于《Systems, Man and Cybernetics》，2006年SMC'06 IEEE国际会议，2006年，第4225-4230页。[19] A. Sadeghi、M. Sadeghi、A. Sharifpour、M. Fakhar、Z. Zakariaei和F. Hajati，基于新型深度学习方法的COVID-19潜在诊断应用，《Nature Scientific Reports》，第14卷，第1期，第280页，2024年。[20] F. Shojaiee和F. Hajati，用于掌纹识别的局部组合导数模式，载于2014年第22届伊朗电气工程会议（ICEE），2014年：IEEE，第965-970页。[21] C. J. P. Sopo、F. Hajati和S. Gheisari，DeFungi：显微真菌图像的直接真菌学检查，arXiv预印本arXiv:2109.07322，2021年。[22] A. Tavakolian、F. Hajati、A. Rezaee、A. O. Fasakhodi和S. Uddin，使用优化的并行 Inception进行快速COVID-19与H1N1筛查，《Expert systems with applications》，第204卷，第117551页，2022年。[23] A. Tavakolian、A. Rezaee、F. Hajati和S. Uddin，使用优化的混合深度模型进行再入院和住院时长预测，《Future Internet》，第15卷，第9期，第304页，2023年。[24] S. Wang、H. Lu、A. Khan、F. Hajati、M. Khushi和S. Uddin，一种用于多类别分类的机器学习软件工具，《Software Impacts》，第13卷，第100383页，2022年。[25] M. Karimi和A. Rezaee，使用Meyer小波对亥姆霍兹方程的柯西问题进行正则化，《Journal of Computational and Applied Mathematics》，第320卷，第76-95页，2017年。[26] B. Mohamadzade和A. Rezaee，用于GSM、WiMAX和WLAN应用的紧凑型宽带双套筒单极天线，《Microwave and Optical Technology Letters》，第59卷，第6期，第1271-1277页，2017年。[27] M. Ramezani、M. Amiri Atashgah和A. Rezaee，用于无人机–无人地面车辆覆盖路径规划的容错多智能体强化学习框架，《Drones》，第8卷，第10期，第537页，2024年。[28] A. Rezaee，生成约束自动机测试数据的遗传共生算法，《Applied and Computational Mathematics》，第6卷，第1期，第126-137页，2008年。[29] A. Rezaee，使用遗传算法设计FIR数字滤波器，《ICTACT Journal on Soft Computing》，第1卷，第1期，第18-22页，2010年。[30] A. Rezaee，使用模糊逻辑确定焊接机器人移动电机的PID控制器系数，《Automatic Control and Computer Sciences》，第51卷，第2期，第124-132页，2017年。[31] A. Rezaee，数字手推式贯入仪的设计、制造与评估，《International Journal of Advanced Smart Sensor Network Systems》，第7卷，第1期，第1-10页，2017年。[32] A. Rezaee，移动机器人控制的模型预测，《Transactions on Environment and Electrical Engineering》，第2卷，第2期，第17-22页，2017年。[33] A. Rezaee和M. K. Golpayegani，使用情感学习的冷热系统智能控制，《Electronics and Electrical Engineering》，第18卷，第4期，第26-30页，2012年。[34] A. Rezaee和M. Pajohesh，基于模糊逻辑的悬架系统控制，《Journal of Communications Technology, Electronics and Computer Science》，第6卷，第1-5页，2016年。[35] A. Sadeghi、F. Hajati、A. Rezaee、M. Sadeghi、A. Argha和H. Alinejad-Rokny，3DECG-Net：用于多标签心律失常检测的心电图融合网络，《Computers in Biology and Medicine》，第182卷，第109126页，2024年。[36] H. Taghvaee、S. M. Seyyedi和A. Rezaee，超材料双频吸波器设计，载于第三届伊朗工程电磁学会议，2014年。[37] A. Tavakolian、F. Hajati、A. Rezaee、A. O. Fasakhodi和S. Uddin，优化的并行 Inception源代码：一种快速COVID-19筛查软件，《Software Impacts》，第13卷，第100337页，2022年。[38] E. Gavagsaz、A. Rezaee和H. Haj Seyyed Javadi，使用可扩展简单随机抽样对MapReduce系统中偏斜数据的归约器进行负载均衡，《The Journal of Supercomputing》，第74卷，第7期，第3415-3440页，2018年。[39] A. Rezaee、A. M. Rahmani、A. Movaghar和M. Teshnehlab，抽象模糊推理云服务的形式化进程代数建模、验证与分析，《The Journal of Supercomputing》，第67卷，第2期，第345-383页，2014年。

[40] S. Sarvghad、A. Rezaee 和 F. Masomi, 《设拉子大学预科学生思维风格与自我效能之间的关系》, 2011年。[41] I. Shahramian、E. Akhlaghi、A. Ramezani、A. Rezaee、N. Noori 和 E. Sharafi, 《重型β-地中海贫血患者血清瘦素浓度研究》, 《伊朗儿科血液学与肿瘤学杂志》, 第3卷, 第2期, 第59页, 2013年。[42] I. Shahramian、M. Razzaghian、A. A. Ramazani、G. A. Ahmadi、N. M. Noori 和 A. R. Rezaee, 《重型β-地中海贫血患者肌钙蛋白与血清铁蛋白水平的相关性》, 《国际心血管研究杂志》, 第7卷, 第2期, 第51页, 2013年。