Journal of System Simulation

Volume 33 | Issue 3 Article 22

3-18-2021

Intelligent Genetic Algorithm for Workforce Scheduling Considering Service Level in IT Maintenance Service

Ruiying Chen
1. School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China; ;

Chengtao Wang
2. Wuhan Windoor Information Technology Company. LTD, Wuhan 430040, China; ;

Zhenyuan Liu
1. School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China; ;3. Key Laboratory of Education Ministry for Image Processing and Intelligent Control, Wuhan 430074, China;

Intelligent Genetic Algorithm for Workforce Scheduling Considering Service Level in IT Maintenance Service

Abstract

Abstract: Aiming at resolving the faults quickly and effectively with limited human resource,an IT maintenance service model with the consideration of service level is proposed, A knowledge model is proposed,a *variety of mutation operators and crossover operators are designed*. and the improved *genetic algorithm (IGA)*, *the intelligent genetic algorithm based on knowledge model (KIGA)* and *the adaptive intelligent genetic algorithm based on knowledge model (KAIGA)* are formed. The results show that the adaptive mutation and crossover probability can accelerate the convergence speed of the solution, and the knowledge model can also improve the optimization effect of the solution

Keywords

workforce scheduling, IT maintenance service, service level, intelligent genetic algorithm, knowledge model

第 33 卷第 3 期 系统仿真学报© Vol. 33 No. 3
2021 年 3 月 Journal of System Simulation Mar. 2021

考虑服务水平的 IT 运维人员调度的智能遗传算法

陈芮莹[1]，王承涛[2]，刘振元[1,3]

(1. 华中科技大学 人工智能与自动化学院，湖北 武汉 430074；2. 武汉问道信息技术有限公司，湖北 武汉 430040；
3. 图像信息处理与智能控制教育部重点实验室，湖北 武汉 430074)

摘要：在人力资源受限的情况下，针对如何在 IT 运维服务中使用有限的人力资源，迅速、有效地对发生的故障进行处理，建立了考虑服务水平的 IT 运维人员调度模型。*提出了一种知识模型，设计了多种符合本问题特性的变异、交叉算子，从而形成了改进的遗传算法、基于知识模型的智能遗传算法和基于知识模型的自适应智能遗传算法*。结果表明，自适应变异、交叉概率能加快解的收敛速度，知识模型能加快解的收敛速度并且提高解的优化效果。

关键词：人员调度；IT 运维服务；服务水平；智能遗传算法；知识模型

中图分类号：TP391 文献标志码：A 文章编号：1004-731X (2021) 03-0732-13

DOI: 10.16182/j.issn1004731x.joss.19-0576

Intelligent Genetic Algorithm for Workforce Scheduling Considering Service Level in IT Maintenance Service

Chen Ruiying[1], *Wang Chengtao*[2], *Liu Zhenyuan*[1,3]

(1. School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China;
2. Wuhan Windoor Information Technology Company. LTD, Wuhan 430040, China;
3. Key Laboratory of Education Ministry for Image Processing and Intelligent Control, Wuhan 430074, China)

Abstract: Aiming at resolving the faults quickly and effectively with limited human resource,an IT maintenance service model with the consideration of service level is proposed, A knowledge model is proposed,a *variety of mutation operators and crossover operators are designed*. and the improved *genetic algorithm (IGA)*, *the intelligent genetic algorithm based on knowledge model (KIGA)* and *the adaptive intelligent genetic algorithm based on knowledge model (KAIGA)* are formed. The results show that the adaptive mutation and crossover probability can accelerate the convergence speed of the solution, and the knowledge model can also improve the optimization effect of the solution

Keywords: workforce scheduling; IT maintenance service; service level; intelligent genetic algorithm; knowledge model

引言

IT 运维服务是 IT 行业中重要的一环，如何有效、快速地进行 IT 运维服务也是企业所关注的问题。本文提出了考虑服务水平的 IT 运维人员调度问题，对一段时间内的故障进行调度，这些故障均是通过专业人士认证后的确定性故障，它具有每个员工处理其所需要的时间、规定的完成时间、以及紧急程度和重要程度等属性。另外，对时间段的选择越小，则问题的实时性越高。

本文提出的问题主要与人员调度问题相关，Bergh 等[1]给出了人员调度问题的最新进展。Semra 等[2]考虑了一个具有灵活的员工可用性和灵活的需求服务的员工调度问题，其中考虑了员工

收稿日期：2019-11-05 修回日期：2020-01-15
基金项目：国家自然科学基金(72071087)，中央高校基本科研业务费(HUST：2017KFYXJJ178)
第一作者：陈芮莹(1996-)，女，硕士生，研究方向为服务系统调度与智能计算。E-mail：475639185@qq.com

的多技能和需求的重要程度。Chen 等[3-4]提出了一种基于经验学习的技术人员调度问题，其中每个技术人员拥有多个技能，且每个时段技术人员相应技能的服务时间会随经验的积累发生变化。Irawan 等[5]提出了一种用于海上风电场维护的优化调度模型，并提出了一种基于 Dantzig-Wolfe 分解的算法。Zamorano 等[6]研究了一个多技能技术人员调度问题，目标是确定技术人员的分配和日常路线的规划。在这些文献中，基于 IT 运维背景进行研究的问题较少，且这些问题中主要注重人员的约束，而较少考虑任务的约束，如任务的规定完成时限、任务的重要程度和紧急程度等，另外，在求解目标中也未考虑在规定时间内未完成任务而产生的延迟成本。

另一方面，人员调度问题已经被证明为一个 NP-hard 问题[7]，关于类似 NP-hard 问题的计算方法，有分支定价法、混合整数规划法等精确算法，也有贪婪算法、遗传算法、蚁群算法等启发式方法，还有一些在基础的算法中融入新元素来提高优化结果和加快优化进程的方法。Khalfay 等[8]回顾了解决人员调度问题的各种算法。Zeng 等[9]提出了一种基于分支价格的方法来求解机场地勤人员的调度问题。Mathlouthi 等[10]提出了一种混合整数线性规划模型来解决多属性技术员的路由和调度问题。安晓亭等[11]提出了一种带局部搜索的改进蚁群优化算法来求解多目标资源受限项目调度问题。Mehran[12]使用一种新颖的贪婪式启发式方法解决了大型航空公司的任务与地勤人员调度问题。邢立宁等[13]提出了一种智能遗传算法，该算法可根据当前的优化结果智能地选择交叉、变异算子。

本文提出考虑服务水平的 IT 运维人员调度问题，与经典的人员调度问题的区别在于其考虑了运维服务的紧急性和重要性，将其作为衡量服务水平的标准，在目标中加入了在规定时间内未完成任务而产生的延迟成本，并且考虑了运维人员的多技能性，多技能人员的高效利用是服务组织的核心[14]。为求解此问题，在传统遗传算法的基础上引入了自适应的交叉、变异概率和在求解过程中学习到的算子知识两种智能机制，并且基于问题特性设计了多种交叉、变异算子，以期能加快遗传算法的收敛速度。

1 问题描述与建模

1.1 问题描述

考虑服务水平的 IT 运维人员调度问题可抽象为：在一定周期内会出现一定数量的故障，有多个运维人员可供管理者选择来进行故障处理，不同的运维人员对于每一个故障的处理时间不同。每个故障都存在一个规定的完成时限，超出时限会产生惩罚成本。故障的紧急程度分为 3 种：紧急、严重、一般，分别对应不同的惩罚成本。提出故障的人员根据级别不同分为 3 个层次：经理、主管、员工，分别对应不同的惩罚系数。每个运维人员的单位时间成本不同。问题的目标是使得处理一系列故障的总成本(薪水成本+惩罚成本)最小，总成本越小意味着服务水平越高。

为了方便建模与求解，特作以下几点假设：

(1) 时间周期中，相邻时刻间的时长相等；

(2) 所有运维人员的最长工作时间都为 T；

(3) 若某个运维人员不能处理某个故障，则该运维人员对此故障的完成时间设置为一个极大值；

(4) 运维人员在一个时刻只能处理一个故障；

(5) 每个故障只需要一位运维人员来处理；

(6) 故障的各个属性参数值已由专业人士确认后提供；

(7) 故障一旦开始处理便不可拆分；

(8) 故障没有紧前紧后关系；

(9) 所有的故障必须在时间周期内被处理。

1.2 数学模型

对考虑服务水平的 IT 运维人员调度问题模型的相关集合和参数作如下定义，如表 1~2 所示。

表 1 集合定义
Tab. 1 Set list

符号	集合
M_α	故障集合，M_α={1,2,…,M}，$m\in M_\alpha$，m 为故障编号
S_α	运维人员集合，S_α={1,2,…,S}，$s\in S_\alpha$，s 为运维人员编号
T_α	时刻集合，T_α={1,2,…,T}，$t\in T_\alpha$，t 为时刻编号

表 2 参数定义
Tab. 2 Parameter list

符号	参数
C_{sm}	故障 m 被运维人员 s 处理所需的时间。反映了运维人员的多技能水平，当其为一个极大值时，表示运维人员 s 无法处理故障 m
T	每位运维人员的最大工作时间
sal_s	运维人员 s 的单位时间成本
SLA_m	故障 m 规定的完成时限
π_m	故障 m 的单位惩罚成本，与故障的紧急程度有关
b_m	故障 m 的单位惩罚系数，与提出故障的人员级别有关

为更好的表述模型，这里对中间变量和决策变量作如下定义：

ST_m 为故障 m 的开始时间；

FT_m 为故障 m 的实际完成时间；

X_{sm} 为运维人员 s 处理故障 m 的总时间，当运维人员 s 未处理故障 m 时，为 0；

y_{smt}=1 为运维人员 s 在 t 时刻正在处理故障 m，否则，为 0；

z_{sm}=1 为运维人员 s 处理了故障 m，否则，为 0。

模型的目标是使薪水成本和惩罚成本之和最小。

$$\min Z=\sum_{s=1}^{S}\left(SaL_s\sum_{m=1}^{M}X_{sm}\right)+\sum_{m=1}^{M}\pi_m b_m\max\left(0,FT_m-SLA_m\right),\tag{1}$$

S.t.

$$ST_m=\min_{s\in S_\alpha,t\in T_\alpha}\{t\times y_{smt}\},\forall m,\tag{2}$$

$$FT_m=\max_{s\in S_\alpha,t\in T_\alpha}\{t\times y_{smt}\}\quad\forall m,\tag{3}$$

$$z_{sm}=\begin{cases}1,\sum_{t=1}^{T}y_{smt}\geqslant 1\\0,\text{否则}\end{cases},\forall s,m,\tag{4}$$

$$FT_m=ST_m+\sum_{s=1}^{S}Q_{sm}z_{sm}\quad\forall m,\tag{5}$$

$$\sum_{m=1}^{M}y_{smt}\leqslant 1,\forall s,t,\tag{6}$$

$$\sum_{s=1}^{S}y_{smt}\leqslant 1,\forall m,t,\tag{7}$$

$$\sum_{s=1}^{S}z_{sm}=1,\forall m,\tag{8}$$

$$X_{sm}=\sum_{t=1}^{T}y_{smt},\forall s,m,\tag{9}$$

$$\sum_{m=1}^{M}X_{sm}\leqslant T,\forall s,\tag{10}$$

$$ST_m,FT_m\in N,\forall m,\tag{11}$$

$$X_{sm}\in N,\forall s,m,\tag{12}$$

$$y_{smt}\in\{0,1\},\forall s,m,t,\tag{13}$$

$$z_{sm}\in\{0,1\},\forall s,m.\tag{14}$$

目标公式(1)表示处理完所有故障的总成本最小；约束公式(2),(3)表示故障开始处理的时刻和故障实际处理完成的时刻；约束公式(4)表示分配给每个运维人员的故障；约束公式(5)表示故障一旦开始处理便不可停止；约束公式(6)表示每个运维人员在单位时间内最多只能处理一项故障；约束公式(7)表示每个故障在单位时间内最多只能由一个运维人员处理；约束公式(8)表示所有故障都必须被处理且只处理一次；约束公式(9)表示每个故障被运维人员处理的时长；约束公式(10)限制了运维人员的最大工作时间，保证了运维人员的工作效率和休息时间；约束公式(11)~(14)表明了变量 ST_m，FT_m，X_{sm}，y_{smt} 和 z_{sm} 的取值范围。

2 求解算法

本文要解决的 IT 运维人员调度问题是一类特殊的人员调度问题，其中增加了多技能约束并且考虑了服务水平，所以本问题也属于 NP-hard 问题，通常在多项式时间内无法解决。遗传算法是一类解决 NP-hard 问题的经典方法，它通过模仿生物进化过程来搜寻近似最优解，具有良好的全局搜索能力且不易较快地陷入局部最优，能在可接受的时间内得到较优的解。

然而，一方面，在传统的遗传算法中，交叉和

变异都只采用一种算子。但在通常情况下，每个算子在不同的实例中表现不一，很难找到能有效求解所有实例的某一种算子。所以，为了提高遗传算法的效率，建立知识模型，采用多种变异和交叉算子来执行遗传算法，尝试挖掘出能有效求解当前实例的算子[15]。另一方面，传统的遗传算法没有考虑进化中个体生存环境的动态变化，因为其交叉、变异概率都采用固定的参数值，当取值较小时会使搜索范围变小，而取值较大时则可能导致已有的优良个体在交叉和变异操作后变得更差。所以，采用自适应算子根据生存环境的动态变化来调整交叉、变异概率。

综上所述，基于本问题特性，本文设计了智能遗传算法来进行求解。其中，设计了 2 种智能算子，一种是知识模型，来提高遗传算法的进化效率，一种是自适应的变异、交叉概率，来使个体更加适应生存环境的变化，以加快算法的收敛速度。

2.1 贪婪算法

贪婪算法是一种求解优化组合问题的经典启发式算法，该方法时间复杂度低，可以在一定的时间内，快速地计算出较优解。

本文贪婪算法规则如下：

$stepGR_1$：根据惩罚成本和惩罚系数将故障排序，惩罚成本高的故障在前，低的在后。当故障惩罚成本相同时，惩罚系数大的故障在前，小的在后。

$stepGR_2$：选择可以处理当前故障的员工，并计算每个员工处理此故障所需要的成本。将此故障分配给处理此故障成本最小的员工。

$stepGR_3$：循环 $stepGR_2$，直到所有故障安排完毕。

2.2 改进的遗传算法

本文提出的改进的遗传算法基本流程与传统的遗传算法相似，不同点在于不再是只有一种变异算子和一种交叉算子，而是设计 4 种变异算子以及 2 种交叉算子，在每一次进行变异、交叉操作时都等概率的来选择变异、交叉算子。

2.2.1 个体

种群中的每个个体代表了一个完整的调度方案。由两段染色体组成，第 1 段为故障序列，第 2 段为人员序列，人员序列依次为处理相对应故障序列中故障的人员编号。编码示例如图 1 所示。

故障序列	3	7	2	1	5	8	4	6
员工序列	1	2	2	1	3	3	1	2

图 1 染色体编码示例

Fig. 1 Example of chromosome coding

2.2.2 初始种群生产

首先，随机生成故障序列，然后，从故障序列的第一个故障开始，找出可以处理此故障的运维人员，并将此故障随机分配给其中一人，如此迭代，直到为所有故障分配了相应的运维人员，以此形成第 2 段染色体。两段染色体组成一个个体，生成满足种群数量的个体。

2.2.3 变异操作

(1) 变异 1

变异 1 为任务对应的资源变异。具体方法为：随机生成一个基因位，改变该基因位故障所对应的运维人员。

(2) 变异 2

变异 2 为运维人员对应的故障顺序变异。具体方法为：随机生成一个运维人员编号，随机选中该运维人员需要处理的 2 个故障，并调换 2 个故障的顺序。

(3) 变异 3

变异 3 为故障顺序的反转逆序变异。具体方法为：随机生成 2 个基因位，将基因位之间的故障序列顺序进行反转，而对应的运维人员不变。

(4) 变异 4

变异 4 为某一运维人员需要处理故障的反转逆序变异。具体方法为：随机选中一个运维人员，将其要处理的故障序列进行反转。

第 33 卷第 3 期 系统仿真学报 Vol. 33 No. 3
2021 年 3 月 Journal of System Simulation Mar. 2021

2.2.4 交叉操作

(1) 单点交叉：对故障染色体进行单点交叉，然后保证人员与故障的对应关系不变来形成第 2 段染色体。

(2) 两点交叉：对故障染色体进行两点交叉，然后保证人员与故障的对应关系不变来形成第 2 段染色体。

2.2.5 选择操作

本文使用的选择算子为轮盘赌选择法。适应度函数设置为 $f_1(x_i)=\frac{\max- f(x_i)}{\max-\min}$，适应度函数的范围是 0~1。max 为种群最大成本，min 为种群最小成本，$f(x_i)$ 为当前染色体成本。由于轮盘赌是一种概率选择法，有可能导致最优的个体没有被遗传到新一代中，所以在此采用精英保留策略，精英个体是每一代中适应度值最高的个体。将每一代的精英个体不经过改变，直接替换掉新一代中适应度最差的个体。

2.3 基于知识模型的智能遗传算法

本文设计了一种知识模型，并将其作为一种智能算子用于遗传算法中。KIGA 与 IGA 基本相同，不同点在于，IGA 中对算子的选择是等概率的，而在 KIGA 中，算子的被选概率是根据知识模型计算得来的。

本文引入的知识模型为算子知识[16]，算子知识主要是指各个算子优化绩效的一种累积知识，也被称为算子绩效知识(Performance Knowledge of Operators，PKO)。在求解当前实例的过程中，当采用 k 算子执行单次操作时，此算子的使用次数 $A(k)$将增加一次，若经过此次操作后，新个体优于原个体，则认为该算子 k 完成的此次操作是成功的，算子 k 的成功次数 $S(k)$加 1。本文的知识模型采用公式(11)来计算算子 k 的优化绩效 $N(k)$，算子绩效每进化一代更新一次，其中 $N'(k)$表示上一代的优化绩效。根据每个算子的优化绩效，使用公式(12)来计算算子的被选概率 $P(k)$。其中，n 表示算子的个数，$P'(k)$表示算子 k 的最低被选概率，它可以保证在后期不管算子绩效如何，每一个算子依然有被选择到的可能。

$$N(k)=\frac{1}{2}(N'(k)+\frac{s(k)}{A(k)}), \tag{15}$$

$$P(k)=P'(k)+(1-\sum_{k=1}^{n}P'(k))\frac{N(k)}{\sum_{k=1}^{n}N(k)}. \tag{16}$$

2.4 基于知识模型的自适应智能遗传算法

本文还将知识模型和自适应公式两种智能算子相结合，同时用于遗传算法中，与 KIGA 的不同点为，KIGA 中的变异、交叉概率为一个常数，而在 KAIGA 中，变异、交叉概率为自适应调整。其基本流程如图 2 所示。

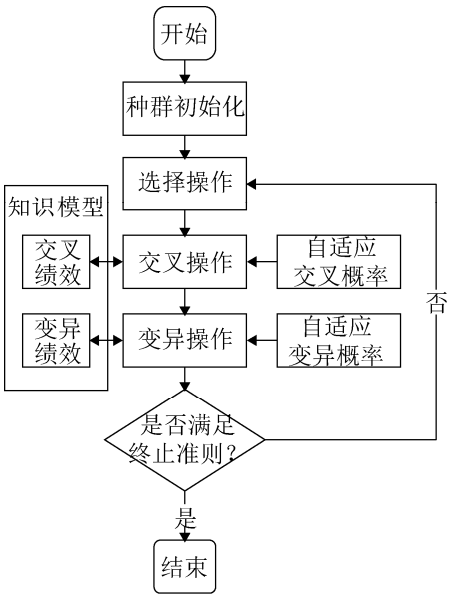


图 2 基于知识模型的自适应遗传算法流程图
Fig. 2 Flow chart of adaptive genetic algorithm based on knowledge model

2.4.1 算法主体框架

基于知识模型的自适应智能遗传算法主框架如算法 1 所示。其基本步骤与一般的遗传算法类似，不同点在于每完成一次迭代，需要使用知识模型来更新各个算子的优化绩效。设 PL 为种群，PS 为预先设置的种群大小，PL'为变异后的种群，PL''为交叉后的种群。

Algorithm 1 Mainframe

1: $PL \leftarrow$ Initialize()

2: initialize the performance knowledge of operators

3: **while** termination criterion not fulfilled **do**

4: $PL' \leftarrow$ Mutation(PL)

5: $PL'' \leftarrow$ Crossover(PL)

6: $PL \leftarrow$ Roulette_selection($PL \cup PL' \cup PL'' \cup PS$)a

7: updating the performance knowledge of operators

8: **end while**

9: **return** PL

2.4.2 自适应变异和交叉概率

本文采用任子武[17]提出的自适应公式(17)~(18)。

$$Pc = \begin{cases} Pc_2 - \dfrac{(Pc_2 - Pc_1)(F' - \overline{F})}{F_{\max} - \overline{F}}, F' \geqslant \overline{F} \\ Pc_2, F' < \overline{F} \end{cases}, \quad (17)$$

$$Pm = \begin{cases} Pm_2 + \dfrac{(Pm_2 - Pm_1)(F - \overline{F})}{F_{\max} - \overline{F}}, F \geqslant \overline{F} \\ Pm_2, F < \overline{F} \end{cases}, (18)$$

式(17)~(18)中：Pc 为交叉概率；F'为两个交叉个体中较大的适应值；$\overline{F}$ 为种群的平均适应值；$F_{\max}$ 为当前种群中的最大适应值；Pm 为变异概率；F 为当前个体的适应度值；Pc_1 和 Pc_2 分别为交叉概率可取范围的最小值和最大值；Pm_1 和 Pm_2 分别为变异概率可取范围的最小值和最大值。

该公式能使适应度较大的个体拥有较小的变异和交叉概率，并使其不为 0，即使优良个体不处于一种停滞状态，从而使算法跳出局部最优解；同时使适应度较小的个体拥有较大的交叉和变异概率，从而尽快提高适应性。

2.4.3 变异操作

对每一代种群 PL 的变异操作如算法 2，所示表示种群 PL_d 中第 d 个个体；PS 为种群规模；Pm 为变异概率；$\overline{F}$ 为种群的平均适应值；$F_{\max}$ 为当前种群中的最大适应值；F 为当前个体的适应度值；Pm_1 和 Pm_2 分别为变异概率可取范围的最小值和最大值。在每一代种群中，使用轮盘赌法得到一个个体，计算其变异概率，若生成的随机概率符合变异概率，则进行个体的变异操作。

Algorithm 2 Mutation

1:**while** d<PS

2: $PL_1 \leftarrow$ Roulette_selection(PL)

3: $F = PL_1\_fitness$

4: **if** $F' > \overline{F}$ **then**

5: $Pm = Pm_2 + \dfrac{(Pm_2 - Pm_1)(F - \overline{F})}{F_{\max} - \overline{F}}$

6: **else**

7: $Pm = Pm_2$

8: **end if**

9: $Pm' = \text{random}(0,1)$

10: **if** $Pm' > Pm$ **then**

11: $PL_d^{'} \leftarrow$ individual_mutation(PL_1)

12: d++

13: **end if**

14: **end while**

15: **return** PL'

对每一个个体的交叉操作需要依据知识模型来进行，如算法 3 所示，每个变异算子都有优化绩效 $Nm(i)$，其计算见公式(11)，根据优化绩效可以计算出每个变异算子的被选概率 $Pm(i)$，其计算见公式(12)，然后，依据此概率来选择变异算子，变异完成后，需要更新算子的使用次数 $Am(i)$，根据是否变异成功来更新算子的成功次数 $Sm(i)$。

Algorithm 3 Individual_mutation

1: $P = \text{random}(0,1)$

2: $Mutation\_operator\_number$=operator_select(P)

3: $PL_m = \text{mutation\_operation}(PL_1, Mutation\_operator\_number)$

4: $Am(Mutation\_operator\_number)$++

5: **if** $PL_m\_fitness > PL_1\_fitness$ **then**

6: $Sm(Mutation\_operator\_number)$++

7: **end if**

8: **return** PL_m

DOI: 10.16182/j.issn1004731x.joss.19-0576

2.4.4 交叉操作

对每一代种群 PL 的交叉操作如算法 4 所示，Pc_1 和 Pc_2 分别为交叉概率可取范围的最小值和最大值；Pc 为交叉概率；F'为 2 个交叉个体中较大的适应值；$\overline{F}$ 为种群的平均适应值；$F_{\max}$ 为当前种群中的最大适应值。在每一代种群中，使用轮盘赌法得到父本和母本后，计算相应的交叉概率，若生成的随机概率符合交叉概率，则进行个体的交叉操作。

Algorithm 4 Crossover

1: **while** d<PS
2: $PL_1 \leftarrow$ Roulette_selection(PL)
$PL_2 \leftarrow$ Roulette_selection(PL)
3: **if** $PL_1\_fitness > PL_2\_fitness$ **then**
4: $F' = PL_1\_fitness$
5: **else**
6: $F' = PL_2\_fitness$
7: **end if**
8: **if** $F' > \overline{F}$ **then**
9: $Pc = Pc_2 - \dfrac{(Pc_2 - Pc_1)(F' - \overline{F})}{F_{\max} - \overline{F}}$
10: **else**
11: $Pc = Pc_2$
12: **end if**
13: $Pc' = \text{random}(0,1)$
14: **if** $Pc' > Pc$ **then**
15: $PL''_d, PL''_{d+1} \leftarrow$ individual_rossover(PL_1, PL_2)
16: d++
17: **end if**
18: **end while**
19: **return** PL''

对每一个个体的交叉操作需要依据知识模型来进行，如算法 5 所示，每个交叉算子都有优化绩效 $Nc(i)$，其计算见公式(17)，根据优化绩效可以计算出每个交叉算子的被选概率 $Pc(i)$，其计算见公式(18)，然后，依据此概率来选择交叉算子，交叉完成后，需要更新算子的使用次数 $Ac(i)$，根据是否交叉成功来更新算子的成功次数 $Sc(i)$。

Algorithm 5 Individual_crossover

1: $P = \text{random}(0,1)$
2: Crossover_operator_number=operator_select(P)
3: $PL_{c1}, PL_{c2} =$ crossover_operation(PL_1, $PL_2, Crossover\_operator\_number$)
4: $Ac(Crossover\_operator\_number)$++
5: **if** $PL_{c1}\_fitness > PL_1\_fitness$ && $PL_{c1}\_fitness > PL_2\_fitness$ && $PL_{c2}\_fitness > PL_1\_fitness$ && $PL_{c2}\_fitness > PL_2\_fitness$ **then**
6: $Sc(Crossover\_operator\_number)$++
7: **end if**
8: **return** PL_{c1}, PL_{c2}

3 计算实验及分析

3.1 案例设计

本节结合某公司实际案例，整理形成了符合本问题特点的不同规模的数据集。计算实验使用 JAVA 编程实现，在 Eclipse 平台上运行，运行环境为 Windows 10(CPU 1.60GHz，8GB 内存)。为比较所提出 4 种算法的优劣性，本文共设计 11 组运维人员数量不同、故障数量不同的案例进行实验。具体参数如表 3 所示。

表 3 案例库参数
Tab. 3 Case library parameter

案例编号	运维人员数量/个	故障数/个
3-15	3	15
4-21	4	21
5-33	5	33
8-55	8	55
10-72	10	72
14-118	14	118
16-136	16	136
18-156	18	156
20-180	20	180
24-218	24	218
25-232	25	232

3.2 参数设置

根据不同规模案例的收敛速度不同，所以在不

同规模下应选取不同的迭代次数，如表 4 所示。KIGA 中每个算子的最低被选概率都为 0.1，初始绩效都为 1。

另外，在遗传算法中，种群规模、交叉概率和变异概率的不同会对实验结果产生一定的影响，所以本处采用正交实验[18]在一定算例下，得到算法参数的最优组合。首先，确定因素水平表，如表 5 所示，其次选择 $L_{25}(5^6)$ 正交表，并以结果值与已知结果值中最好的值之间的偏差作为评判指标进行实验，首先在案例 4-21 下进行实验，如图 3 所示。找出偏差最小的参数组合，若无法得到唯一一组，则继续在案例 5-33、8-55 下进行实验，得到的结果如表 6，其中‘—’表示已无需进行实验。

表 4 各个规模下算法的迭代次数

Tab. 4 The number of iterations of the algorithm at various scales

案例编号	迭代次数	案例编号	迭代次数
3-15	400	16-136	1 000
4-21	400	18-156	1 000
5-33	400	20-180	1 500
8-55	400	24-218	1 500
10-72	1 000	25-232	1 500
14-118	1 000		

表 5 水平因素表

Tab. 5 Table of horizontal factors

水平	因素		
	种群规模	交叉概率	变异概率
1	20	0.5	0.01
2	40	0.6	0.05
3	60	0.7	0.1
4	80	0.8	0.2
5	100	0.9	0.3

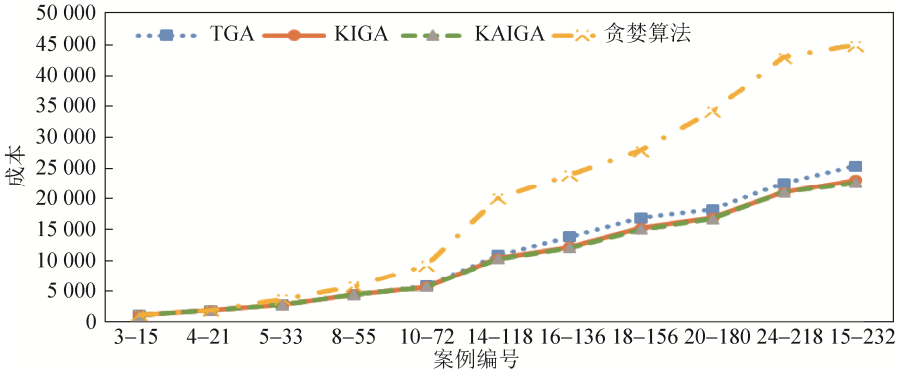


图 3 贪婪算法、IGA、KIGA 和 KAIGA 解的均值分布图

Fig. 3 Mean distribution of Greedy algorithm, IGA, KIGA and KAIGA solutions

表 6 参数的鉴定实验

Tab. 6 Parameter identification experiment

试验号	种群规模	交叉概率	变异概率	IGA			KIGA		
				4-21	5-33	8-55	4-21	5-33	8-55
1	1(20)	1(0.5)	1(0.01)	0.002 1	—	—	0.002 1	—	—
2	1(20)	2(0.6)	2(0.05)	0	0.016 6	—	0	0.018 0	—
3	1(20)	3(0.7)	3(0.10)	0.002 1	—	—	0.009 6	—	—
4	1(20)	4(0.8)	4(0.20)	0.002 1	—	—	0.002 1	—	—
5	1(20)	5(0.9)	5(0.30)	0.004 8	—	—	0.004 8	—	—
6	2(40)	1(0.5)	2(0.05)	0	0.003 6	—	0	0	0
7	2(40)	2(0.6)	3(0.10)	0.002 1	—	—	0	0.014 4	—
8	2(40)	3(0.7)	4(0.20)	0.002 1	—	—	0	0.013 3	—
9	2(40)	4(0.8)	5(0.30)	0.002 1	—	—	0.004 8	—	—
10	2(40)	5(0.9)	1(0.01)	0	0.002 9	—	0	0	0.001 2
11	3(60)	1(0.5)	3(0.10)	0	0	0.0279	0	0.002 5	—
12	3(60)	2(0.6)	4(0.20)	0	0.003 6	—	0.002 1	—	—
13	3(60)	3(0.7)	5(0.30)	0	0.013 7	—	0	0	0.045 3
14	3(60)	4(0.8)	1(0.010)	0	0.010 8	—	0.009 6	—	—
15	3(60)	5(0.9)	2(0.05)	0	0.002 5	—	0.004 8	—	—
16	4(80)	1(0.5)	4(0.20)	0	0	0	0	0.002 5	—

第 33 卷第 3 期 系统仿真学报 Vol. 33 No. 3
2021 年 3 月 Journal of System Simulation Mar. 2021

续表

试验号	种群规模	交叉概率	变异概率	IGA			KIGA		
				4-21	5-33	8-55	4-21	5-33	8-55
17	4(80)	2(0.6)	5(0.30)	0	0.013 3	—	0	0	0.001 6
18	4(80)	3(0.7)	1(0.01)	0.002 1	—	—	0	0	0.042 1
19	4(80)	4(0.8)	2(0.05)	0	0.003 6	—	0	0.144 0	—
20	4(80)	5(0.9)	3(0.10)	0	0.013 0	—	0.002 1	—	—
21	5(100)	1(0.5)	5(0.30)	0	0.002 5	—	0	0.002 5	—
22	5(100)	2(0.6)	1(0.01)	0	0.002 5	—	0.002 1	—	—
23	5(100)	3(0.7)	2(0.05)	0.0021	—	—	0.004 8	—	—
24	5(100)	4(0.8)	3(0.10)	0	0.003 0	0.0150	0	0	0.015 0
25	5(100)	5(0.9)	4(0.2)	0	0.002 5	0.0177	0	0	0.017 7

表 6 可得，在 IGA 中，最优参数组合为：种群规模为 80；交叉概率为 0.5；变异概率为 0.2，在 KIGA 中，最优参数组合为：种群规模为 40；交叉概率为 0.5；变异概率为 0.05。由于 KAIGA 中交叉概率和变异概率是自适应的，则使交叉概率在 0.5~0.9 之间，变异概率在 0.01~0.3 之间，与正交实验中的交叉概率、变异概率因素的最大最小水平一致，还在案例 5-33 下对不同的种群规模因素进行了实验，如图 3 所示。如表 7 所示，最终 KAIGA 的种群规模设置为 60。

表 7 KAIGA 种群规模对比实验

Tab. 7 Comparison experiment of KAIGA population size

种群规模	偏差	种群规模	偏差
20	0.015 5	80	0.010 8
40	0.002 5	100	0.010 8
60	0		

3.3 结果分析

3.3.1 结果对比

(1) 均值对比

本处对 4 种算法在各个案例规模下解的均值和计算时间进行了统计，如表 8 所示。其中，实验终止条件为表 4 所示的固定代数，表中的比例表示各个算法相对于贪婪算法的优化比例。图 3 为解的均值分布图。由表 8 和图 3 可知，提出的 3 种算法在每一个案例上都远远优于贪婪算法，对于提出的 3 种算法，在案例规模较小时差别不大，但随着案例规模增大，加入了知识模型的 KIGA 和 KAIGA 算法要明显优于 IGA，但加入了自适应变异、交叉概率的 KAIGA 与 KIGA 没有较大差别，说明知识模型对解的优化有一定的效果，而自适应变异、交叉概率对解的优化基本没有效果。另外，在计算时间上，3 种算法在同一个案例下的计算时间差别不大，但因为不同案例规模下收敛的代数不同，所以在案例规模变大时，迭代的次数有所增加，所以时间增长较多。

(2) 假设检验

PD(the percentage deviation，偏差百分比)代表了当前解离最优解的距离。设 3 种算法 A，B，C 在同一案例下运行 n 次得到的解分别为 $\{E_{A1}, E_{A2}, \ldots, E_{An}\}$，$\{E_{B1}, E_{B2}, \ldots, E_{Bn}\}$，$\{E_{C1}, E_{C2}, \ldots, E_{Cn}\}$。记 θ 为 $3n$ 个解中的最优值。可得到 3 种算法的偏差分布分别为：a=$\{PD_{A1}, PD_{A2}, \ldots, PD_{An}\}$，b=$\{PD_{B1}, PD_{B2}, \ldots, PD_{Bn}\}$，c=$\{PD_{C1}, PD_{C2}, \ldots, PD_{Cn}\}$，其中 $PD_{il}=(E_{li}-\theta)/\theta$, E_{li} 为算法 l 第 i 次运行的评价值，PD_{il} 为算法 l 在第 i 次运行距最优值的偏差。

为了进一步验证算法的性能，对 3 种算法的偏差在 0.05 的显著性水平下进行秩和检验。此处的 n 设置为 10 次。表 9 统计了 3 种算法在各个案例下的偏差均值。检验结果的表现形式如下：‘+’表示 KIGA 优于该算法；‘–’表示 KIGA 劣于该算法；‘≈’表示 KIGA 与该算法没有显著性差异。当存在显著性差异时，根据偏差均值来对算法进行比较。

由表 9 可知，在案例较小时，IGA、KIGA 和 KAIGA 的 PD 值无显著性差异，但随着案例的增大，加入知识模型的 KIGA 的 PD 值显著优于 IGA，而加入自适应变异、交叉概率的 KAIGA 与 KIGA 的 PD 值，除了案例 18-156 外，没有显著性差异。此结论与均值比较中得出的结论一致。

表 8 4 种算法的结果均值

Tab. 8 Mean value of results of the four algorithms

案例	贪婪算法		IGA			KIGA			KAIGA		
编号	成本	计算时间/s	成本	计算时间/s	优化比例/%	成本	计算时间/s	优化比例/%	成本	计算时间/s	优化比例/%
3-15	1 194	0.020	1 153	37.96	3.43	1 153	35.99	3.43	1 156	36.82	3.18
4-21	1 965	0.023	1 881	40.38	4.27	1 879	37.59	4.38	1 881	36.76	4.27
5-33	3 776	0.032	2 794	44.48	26.00	2 784	41.80	26.27	2 794	41.37	26.01
8-55	5 794	0.057	4 474	61.05	22.78	4 389	61.26	24.25	4 438	61.86	23.40
10-72	9 248	0.061	5 984	200.56	35.29	5 776	204.20	37.54	5 742	202.23	37.91
14-118	20 114	0.101	10945	744.07	45.59	10326	726.36	48.66	10 411	731.48	48.24
16-136	23972	0.125	13203	1 106.57	44.92	12154	1 103.61	49.30	12 333	1 097.80	48.55
18-156	27901	0.145	16863	1 913.62	39.56	15047	1 868.02	46.07	15 245	1 873.35	45.36
20-180	34230	0.185	18067	4 130.65	47.22	16776	4 148.71	50.99	17 012	4 136.39	50.30
24-218	43039	0.210	23136	7 614.23	46.24	20989	7 003.30	51.23	20 953	7 125.86	51.32
25-232	45007	0.235	25589	8 268.88	43.14	22785	7 839.08	49.37	22 703	8 112.18	49.56

注：优化比例表示该算法相对于贪婪算法的平均优化比例

表 9 不同规模下 3 种算法的偏差均值

Tab. 9 Mean deviations of the three algorithms at different scales

案例编号	KIGA	IGA	KAIGA
3-15	0.000 0	(≈)0.000 0	(≈)0.002 4
4-21	0.001 3	(≈)0.000 0	(≈)0.001 3
5-33	0.002 2	(≈)0.005 6	(≈)0.005 9
8-55	0.016 0	(+)0.051 9	(≈)0.027 4
10-72	0.022 7	(≈)0.015 0	(≈)0.012 8
14-118	0.011 6	(+)0.072 2	(≈)0.019 9
16-136	0.006 1	(+)0.092 8	(≈)0.020 8
18-156	0.004 6	(+)0.125 9	(+)0.017 8
20-180	0.064 5	(+)0.146 4	(≈)0.079 4
24-218	0.013 8	(+)0.117 5	(≈)0.012 1
25-232	0.0155	(+)0.1405	(≈)0.0119

3.3.2 IGA 和 KIGA 对比

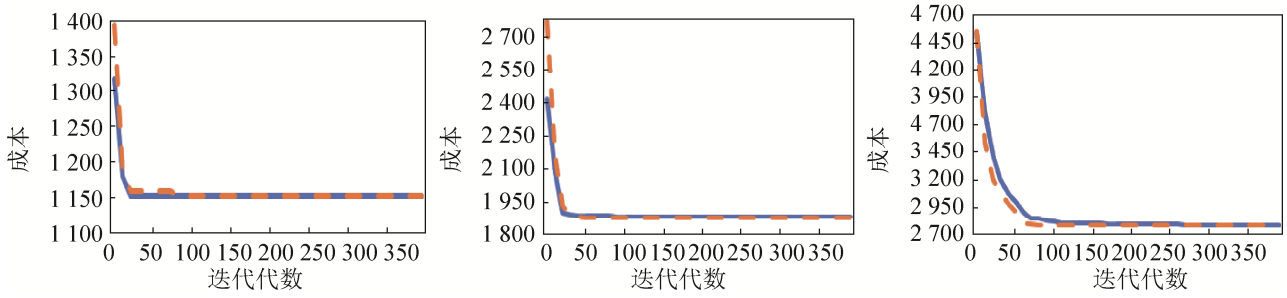
本处对 IGA 和 KIGA 2 种算法在各个案例下的收敛速度进行分析，其中对每个案例运行 10 次，并对结果取平均值，得到每一代的最优个体随迭代次数的变化如图 4。

由图 4 可知，在案例较小时，KIGA 算法的优越性不明显。但随着案例规模的增大，加入了知识模型后的 KIGA 能更快的收敛到最优解，且得到的解优于 IGA。

3.3.3 KIGA 和 KAIGA 对比

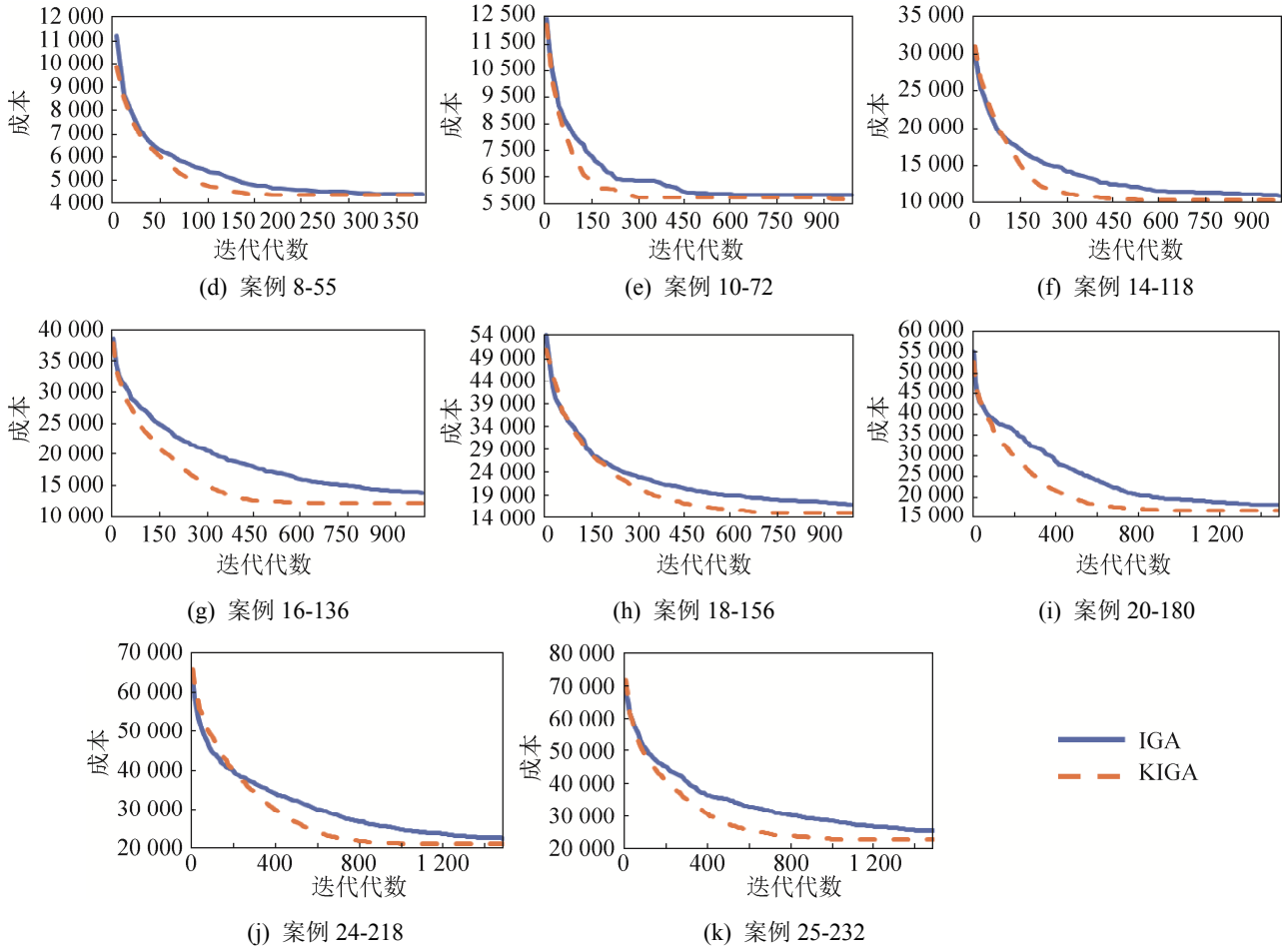
本处对 KIGA 和 KAIGA 2 种算法在各个案例下的收敛速度进行分析，其中对每个案例运行 10 次，并对结果取平均值，得到每一代的最有个体随迭代次数的变化如图 5。

由图 5 可知，在案例较小时，前两个案例中，KAIGA 与 KIGA 基本无差别，但随着案例规模的增大，在案例 5-33 和案例 8-55 中，KIGA 的收敛速度要优于 KAIGA，当案例规模为 10-72，甚至更大时，加入了自适应变异、交叉概率的 KAIGA 的收敛速度明显优于 KIGA。



(a) 案例 3-15 (b) 案例 4-21 (c) 案例 5-33

第 33 卷第 3 期 系统仿真学报 Vol. 33 No. 3
2021 年 3 月 Journal of System Simulation Mar. 2021



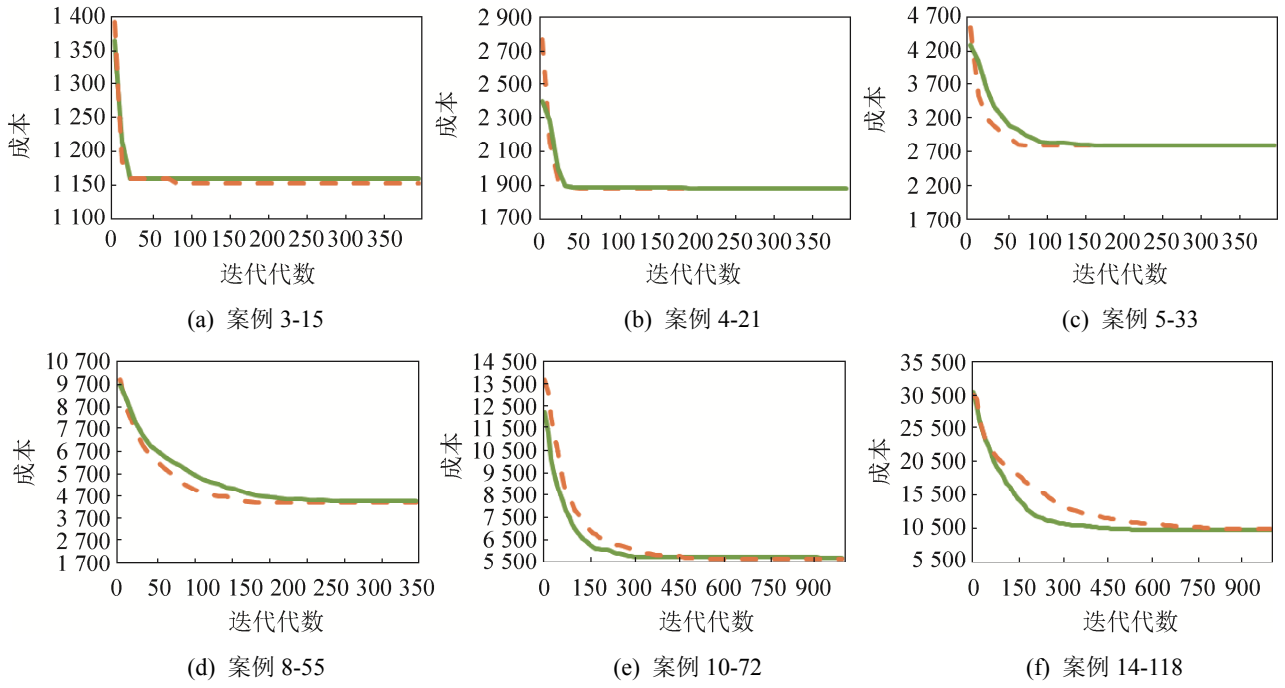
(d) 案例 8-55 (e) 案例 10-72 (f) 案例 14-118

(g) 案例 16-136 (h) 案例 18-156 (i) 案例 20-180

(j) 案例 24-218 (k) 案例 25-232

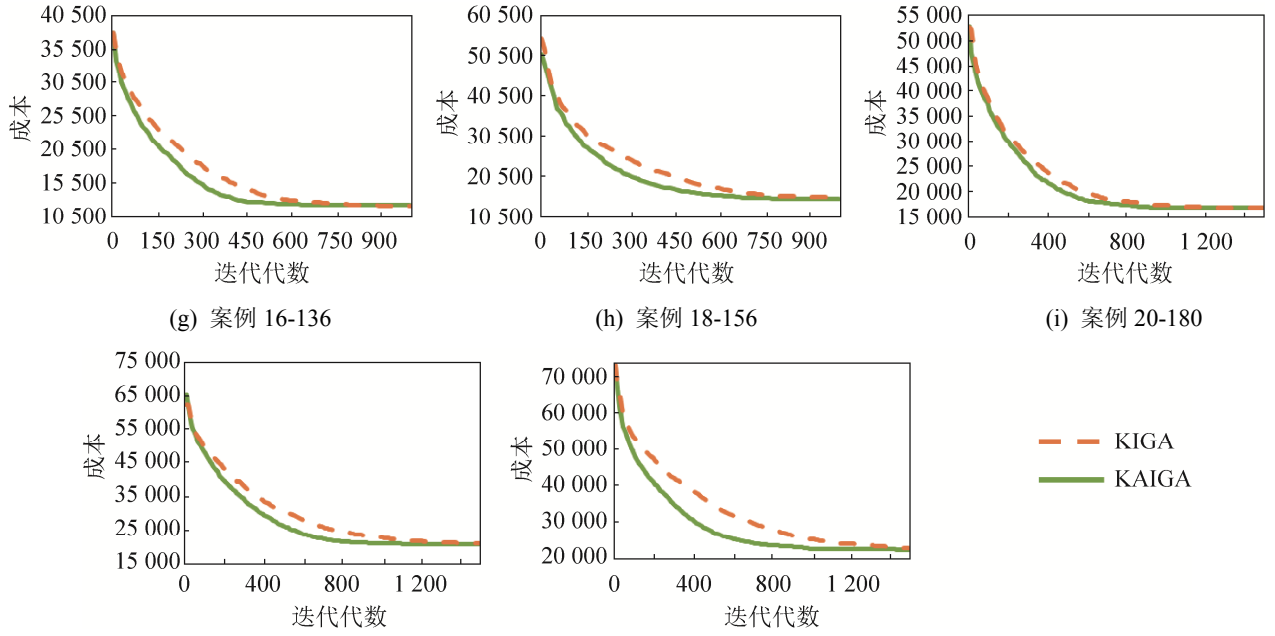
图 4 IGA 和 KIGA 在不同案例下的收敛速度

Fig. 4 Convergence rates of IGA and KIGA in different cases



(a) 案例 3-15 (b) 案例 4-21 (c) 案例 5-33

(d) 案例 8-55 (e) 案例 10-72 (f) 案例 14-118



(g) 案例 16-136 (h) 案例 18-156 (i) 案例 20-180

(j) 案例 24-218 (k) 案例 25-232

图 5 KIGA 和 KAIGA 在不同案例下的收敛速度

Fig. 5 Convergence rates of KIGA and KAIGA in different cases

4 结论

本文对考虑服务水平的 IT 运维人员调度问题进行描述和建模，基于问题特性设计了多种交叉、变异方式同时用于改进的遗传算法中，并设计了 2 种智能算子，分别是知识模型和自适应交叉、变异概率，将其融入改进的遗传算法中，形成了 2 种智能遗传算法。最后分别用 3 种算法对问题进行求解。实验结果表明：在案例规模较小时，3 种算法的性能差异不大，但随着案例规模的增大，知识模型能有效的加快解的收敛速度，并且使得到的解更优，而自适应的变异、交叉概率能加快解的收敛速度，但在解的优化上没有明显效果。

针对现实中的 IT 运维服务，首先，一个故障可能需要两人或多人同时进行维修，且是经常发生的情况，所以考虑多人协同运维是必要的；其次，在运维服务中，实时性也是一个重要的服务水平指标，所以在之后的工作中可以考虑对实时的故障进行动态调度或进行重调度；最后，一个运维人员可能需要在一天内往返于不同的维修地点，把路径规划考虑其中也可以优化运维服务。算法上，可以使用学习型算子来结合其它启发式算法，如粒子群算法、模拟退火算法等，形成新的学习型智能优化算法来进行求解。

参考文献：

[1] Bergh J, Beliën J, Bruecker P. Personnel Scheduling: A Literature Review[J]. European Journal of Operational Research (S0377-2217), 2013, 226(3): 367-385.

[2] Ağralı S, Taşkın Z C, Ünal A T. Employee Scheduling in Service Industries with Flexible Employee Availability and Demand[J]. Omega (S0305-0483), 2016, 66(Part A): 159-169.

[3] Chen X, Thomas B W, Hewitt M. The Technician Routing Problem with Experience-based Service Times[J]. Omega (S0305-0483), 2016, 61(6): 49-61.

[4] Chen X, Thomas B W, Hewitt M. Multi-period Technician Scheduling with Experience-based Service Times and Stochastic Customers[J]. Computers & Operations Research (S0305-0548), 2017, 82(6): 1-14.

[5] Irawan C A, Ouelhadj D, Jones D, et al. Optimisation of Maintenance Routing and Scheduling for Offshore Wind Farms[J]. European Journal of Operational Research (S0377-2217), 2017, 256(1): 76-89.

[6] Zamorano E, Stolletz R. Branch-and-price Approaches for the Multiperiod Technician Routing and Scheduling Problem[J]. European Journal of Operational Research

https://dc-china-simulation.researchcommons.org/journal/vol33/iss3/22
DOI: 10.16182/j.issn1004731x.joss.19-0576

(S0377-2217), 2017, 257(1): 55-68.

[7] Garey M R, Johnson D S. Computers and Intractability: a Guide to the Theory of NP-completeness[M]. New York: W.H. Freeman & Co, 1979.

[8] Khalfay A, Crispin A, Crockett K. A Review of Technician and Task Scheduling Problems, Datasets and Solution Approaches[C]// Intelligent Systems Conference. 2017: 288-296.

[9] Zeng L, Zhao M, Liu Y. Airport Ground Workforce Planning with Hierarchical Skills: A New Formulation and Branch-and-price Approach[J]. Annals of Operations Research (S0254-5330), 2019, 275(1): 1-14.

[10] Mathlouthi I, Gendreau M, Potvin J Y. Mixed Integer Linear Programming for a Multi-attribute Technician Routing and Scheduling Problem[J]. Information Systems & Operational Research (S0315-5986), 2017, 56(1): 1-17.

[11] 安晓亭, 张梓琪. 基于改进蚁群优化的多目标资源受限项目调度方法[J]. 系统工程理论与实践, 2019, 39(2): 509-519.
An Xiaoting, Zhang Ziqi. Multi-objective Resource Constrained Project Scheduling Problem Based on Improved ant Colony Optimization[J]. Systems Engineering-Theory & Practice, 2019, 39(2): 509-519.

[12] Hojati M. A Greedy Heuristic for Shift Minimization Personnel Task Scheduling Problem[J]. Computers & Operations Research (S0305-0548), 2018, 100(12): 66-76.

[13] 邢立宁, 陈英武, 蔡怀平, 等. 求解全局优化问题的智能遗传算法[J]. 系统仿真学报, 2006, 18(4): 1067-1069.
Xing Lining, Chen Yingwu, Cai Huaiping, et al. Intelligent Genetic Algorithm for Global Optimization Problem[J]. Journal of System Simulation, 2006, 18(4): 1067-1069.

[14] 刘振元. 服务系统中的多技能人员调度[M]. 北京: 清华大学出版社, 2019.
Liu Zhenyuan. Multi-skilled Personnel Scheduling in Service System[M]. Beijing: Tsinghua University Press, 2019.

[15] 邢立宁. 演化学习型智能优化方法及其应用研究[D]. 长沙: 国防科学技术大学信息系统与管理学院, 2009.
Xing Lining. Research on the Learnable Intelligent Optimization Approaches and Its Applications[D]. Changsha: School of Information Systems and Management, National University of Defense and Technology, 2009.

[16] 姚锋, 邢立宁, 李菊芳. 求解双层 CARP 优化问题的知识型遗传算法[J]. 系统工程理论与实践, 2014, 34(1): 239-247.
Yao Feng, Xing Lining, Li Jufang, et al. Knowledge-based Genetic Algorithm to the Double Layer Capacitated Arc Routing Problems[J]. Systems Engineering-Theory & Practice, 2014, 34(1): 239-247.

[17] 任子武, 伞冶. 自适应遗传算法的改进及在系统辨识中应用研究[J]. 系统仿真学报, 2006, 18(1): 41-43.
Ren Ziwu, San Ye. Improved Adaptive Genetic Algorithm and Its Application Research in Parameter identification[J]. Journal of System Simulation, 2006, 18(1): 41-43.

[18] 王雷, 蔡劲草, 李明. 基于正交试验的遗传算法参数优化[J]. 南京师范大学学报, 2016, 16(2): 81-85.
Wang Lei, Cai Jingcao, Li Ming. Parameter Optimization of Genetic Algorithm Based on Orthogonal Experiment[J]. Journal of Nanjing Normal University, 2016, 16(2): 81-85.