计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(12):279−287 [doi: 10.15888/j.cnki.csa.008181]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 SMT 的机组排班问题优化求解①

刘锡鹏, 陈　寅

(华南师范大学 计算机学院, 广州 510631)
通讯作者: 陈　寅, E-mail: ychen@scnu.edu.cn

摘　要: 机组排班是航空公司运营计划非常重要的一个环节, 合理的机组排班可以为航空公司省下一大笔机组成本支出, 从而增加航空公司的收益. 由于机组排班过程涉及大量的复杂约束, 属于 NP 难问题, 因此优化求解困难. 本文提出了一种基于可满足性模理论 (Satisfiability Modulo Theories, SMT) 的航空公司机组排班问题的优化求解方法, 将机组排班过程中的各种约束转化为一阶逻辑公式, 设立求解目标为最小化成本和最大化机组利用率, 将问题转化为求在给定逻辑公式可满足情况下的最优解, 并利用 SMT 求解器 Z3 进行求解. 实验表明, 本文的算法能有效的求解一定规模航班计划的机组排班问题, 给航空公司带来一定的收益.

关键词: 可满足性模理论 (SMT); 一阶逻辑; 机组排班; 优化模型; Z3

引用格式: 刘锡鹏,陈寅.基于 SMT 的机组排班问题优化求解.计算机系统应用,2021,30(12):279–287. http://www.c-s-a.org.cn/1003-3254/8181.html

Optimal Solution to Crew Scheduling Problem Based on SMT

LIU Xi-Peng, CHEN Yin

(School of Computer, South China Normal University, Guangzhou 510631, China)

Abstract: Crew scheduling is an important part of an airline operation plan. Reasonable crew scheduling can help airlines save great crew costs and increase their revenue. Since the crew scheduling process involves massive complex constraints, it is an NP-hard problem; thus it is difficult to optimize the solution. This study proposes an optimal solution to crew scheduling problems according to Satisfiability Modulo Theories (SMT), which converts various constraints in the crew scheduling process into first-order logic formulas and sets the solution goal as minimum cost and maximum crew utilization. In addition, it transforms the problem into an optimal solution under the satisfiable condition of the given logic formula and uses the SMT solver, Z3. Experiments show that the algorithm in this study can solve the crew scheduling problem of a certain-scale flight plan, bringing benefits to airlines.

Key words: Satisfiability Modulo Theories (SMT); first-order logic; crew scheduling; optimization model; Z3

1　引言

航空公司运营计划包括航班计划、飞机排班计划和机组排班计划等, 其中机组排班计划是运营计划的重要组成部分. 近年来随着民航业的发展, 航线网络日益复杂, 民航管理局等相关部门为保障飞行安全制定了相关的行业规范, 例如 CCAR-121-R6[1] 等. 由于航线和约束的复杂性, 单纯使用手工模式进行排班是十分困难的, 另一方面, 航班计划在执行过程中难免会遇到由机械故障、天气原因或者机场调度等原因造成的航班变更, 因此在已有计划的基础上灵活地调整航班安排也是一个非常现实的需求. 由于人力成本的上升, 机组人员的收入水平提高, 机组排班的结果直接决定航空公司的实际机组成本支出, 从而影响航空公司的运营收益. 因此, 在满足相关约束下保障飞行安全的同时

① 收稿时间: 2021-02-02; 修改时间: 2021-03-05; 采用时间: 2021-03-16

设计一个良好高效的算法来完成机组排班计划的编排对于航空公司提高运营效率和增加收益有着重要的现实意义.

本文提出一种基于可满足性模理论 (Satisfiability Modulo Theories, SMT) 的机组排班问题的优化求解方式. SMT 在布尔可满足性理论 (SAT) 的基础上加入背景理论 (例如, 算术运算、向量和数组等), 可以适用于更广泛的应用场景. 机组排班问题中既存在一些逻辑约束, 又存在一些数量约束, 因此考虑采用 SMT 来进行求解. 另一方面, 现代的 SMT 求解器都支持增量式的求解 (incremental solving), 即可在求解的任何阶段, 增加或者删除约束, 而无需重新搜索整个状态空间, 这也为航班排班中灵活调整的实现带来的便利.

本文通过对机组排班问题的分析, 建立了机组排班问题的多目标优化模型, 将各种约束表述为一阶逻辑公式, 并利用 SMT 求解器 Z3 进行增量求解, 实现了一定规模的机组排班问题的优化求解.

本论文的组织结构如下: 在第 2 节, 对机组排班问题的定义、已有的研究工作以及 SMT 求解技术进行详细介绍. 在第 3 节, 提出机组排班的数学模型和相应算法. 在第 4 节, 通过实验对航班实例进行求解并对结果进行分析. 在第 5 节也是最后一节, 对全文的工作做出系统性的总结.

2 研究背景

2.1 机组排班问题概述

机组排班计划问题是指在民航管理局相关适航文件和具体航空公司制定的一系列规定的前提下, 将航班计划内的各个航班合理的分配给机组飞行员和乘务人员执行的过程. 在航空公司运营总成本中, 机组成本是仅次于燃料成本的第二大费用[2], 由于机组成本比燃料成本具有更好的可控性, 通过合理的分配可降低一些不必要的额外成本, 为航空公司带来更多的收益. 为了保障飞行安全, 机组排班过程往往涉及大量的约束, 要满足一系列相关法律法规对飞行时间、人员要求等严格规定, 这使得机组排班模型复杂, 优化求解困难, 属于 NP 难问题[3].

机组排班计划问题一般分为两个子问题: 机组排班问题和机组人员指派问题. 其中机组排班问题又称为机组配对问题, 其任务是在满足相关约束条件下生成一系列可由机组执行的从机组基地出发并最终返回基地的航班任务环, 要求覆盖所有待执行的航班, 优化目标是任务环成本的最小化[4]. 机组人员指派问题[5] 的任务是要将机组排班阶段生成的任务环中的任务合理的分配给具体的机组人员执行, 要求考虑任务分配的公平性原则. 本文主要研究机组排班问题.

2.2 已有的研究工作

机组排班问题是复杂的组合优化问题, 通常可将其描述为集合分割问题 (Set Partition Problem, SPP) 或集合覆盖问题 (Set Covering Problem, SCP). 本文将已有的研究工作分为确定性算法和非确定性算法.

非确定性算法方面, Kornilakis 等[6] 采用启发式两阶段求解方法, 先利用深度优先算法寻找所有可行任务环, 然后从所有任务环中选择最小成本的任务环组合, 选择了遗传算法来进行航班的组合优化. 赵天洋[7] 对机组排班理论和算法进行分析, 比较了常用的遗传算法、蚁群算法和模拟退火算法的优缺点, 对机组排班各阶段进行分析设计, 将机组排班过程分为勤务生成、机组排班优化和机组人员指派 3 个部分, 采用改进的遗传算法进行优化求解. 吴苏阳[8] 考虑机组排班中任务环组环的质量和效率, 从航空公司运营成本、生产计划的稳定性、生产计划质量和飞行时间均衡方面对飞行任务的组环进行优化, 采用遗传算法求解. Aggarwal 等[9] 提出了一种用于大规模复杂航班网络生成初始航班任务环的启发式算法, 采用分割和覆盖方法, 从大规模航班数据中随机分割部分航班生成任务环, 然后将优化求解后的任务环加入初始航班任务环 (IFS) 中, 直到所有航班被覆盖. 张文成等[10] 提出改进二进制粒子群算法用于机组配对优化, 以机组利用率最大为优化目标, 利用深度优先搜索算法从航班计划表中生成初始航班配对, 在改进二进制粒子群算法寻优过程中引入惩罚因子和自适应权重, 提高了算法的速度和解的质量. Chen 等[11] 整合飞机排班和机组排班问题, 提出多目标组合优化模型, 采用 NSGA-II(带精英策略的非支配排序的遗传算法) 结合修复策略求解, 通过实际数据实验得出的时刻表比专家制定计划更优.

在确定性算法方面, Marsten 等[12] 建立了机组排班问题的集合分割模型, 采用拉格朗日松弛和次梯度优化方法进行求解. Graves 等[13] 采用列生成算法对机组排班问题进行求解, 系统分为生成器和优化器, 生成器负责生成候选航班配对传递给优化器, 优化器负责寻找使总成本最小的一组航班配对, 通过不断迭代至最

优并覆盖所有航班. 蓝伯雄等[14] 在机组排班阶段采用混合集合规划进行高层建模, 利用一阶逻辑和集合推理对机组排班问题进行表述, 将运筹学优化方法与业务逻辑相结合, 建立了贴近实际的机组排班模型. Zeighami 等[15] 集成了机组排班问题和机组人员分配问题, 综合考虑机组成本和飞行员喜好, 提出结合拉格朗日分解、列生成法和动态约束聚合的综合算法, 实验表明该方法能有效节省大量成本和满足机组人员喜好.

2.3 SMT 求解技术介绍

可满足性模理论 (Satisfiability Modulo Theories, SMT)[16] 是指检查给定的在相关背景理论 (如算术、位向量、数组和未解释函数等) 下一阶逻辑公式是否可满足的问题. 给定一个 SMT 公式 F, 如果存在一组赋值使得 F 为真, 则称 F 是可满足的, 否则称 F 是不可满足的. SMT 是一种说明性语言, 可以使用一阶逻辑语言描述问题的约束, 同时支持各种复杂约束, 能很好的描述 NP 及 co-NP 问题, 在优化问题求解[17]、程序验证[18]、静态分析[19] 等领域有突出优势.

SMT 的判定方法一般被称为 SMT 求解器 (SMT solver), 目前主流的 SMT 求解器有 Z3[20]、CVC4、Yices2 等, 这些求解器都能较好的处理大规模工业化问题, 本文采用的是微软的 Z3 作为 SMT 求解器来求解机组排班问题.

Z3 是微软组织开发的 SMT 求解器, 在扩展性、表达能力以及求解效率等方面都较为出色, 是目前最好用的 SMT 求解器之一. Z3 在 Github 发布了开源的项目, 在近年来的 SMT 求解器的竞赛中一直处于领先的地位. Z3 内部采用一种类似于 Lisp 的 SMT-LIBv2 语言[21], 同时也提供了包括 C, C++和 Python 等常用的编程语言接口. 本文采用的 Z3Py 就是 Z3 提供的 Python 开发包, 可以方便约束求解和系统其它部分的接口. Z3 支持多种理论, 不仅可以用于验证多个逻辑公式的可满足性, 也能给出一组满足约束的解; 此外还支持增量求解, 可以在不改变原问题基础上增加新的约束, 而无需重新计算, 加快了求解速度. Z3 采用栈的方式, 可以在新约束不满足时回退到上一个阶段.

机组排班问题本质上是在一系列约束条件下的优化求解问题, 要满足民航局和航空公司制定的关于飞行安全的复杂适航条例, 属于 NP 难问题. SMT 可以通过一阶逻辑公式很好的表达机组排班过程中各种复杂的约束, 在优化求解方面具有突出优势, SMT 求解器 Z3 在扩展性、表达能力以及求解效率等方面都较为出色, 通过设置求解目标, 可以在满足相关约束的情况下高效地求解出问题的最优解.

3 机组排班模型及算法设计

3.1 问题的模型

机组排班问题中机组排班问题通常用集合分割模型表示, 本文以最小化任务环总成本为主优化目标, 以最大化机组利用率为次目标建立多目标优化模型. 其中机组成本包括任务津贴、飞行津贴、在外过夜成本以及加机组成本等, 任务津贴是机组执行任务所要支付的必要成本; 飞行津贴是机组参与飞行时的补贴, 一般与飞行时间有关; 过夜成本是指机组在执行的某一天的任务中, 最后的航班降落机场为非基地机场, 无法回到基地休息而在外过夜产生的旅店成本; 加机组成本是一个机组为了完成某个任务而作为乘客搭乘另一个机组的航班到达目的地执行任务而产生的额外开支. 机组利用率是机组执行任务中有效的工作时间占据的比重, 等于机组飞行时间/机组执勤时间.

假设 P 是所有航班任务环的集合, p 是 P 中一个任务环, c_p是任务环 p 的总成本, x_p是 0-1 变量, 表示任务环是否被选中为最终的任务环, 则主目标函数 $Obj1$如下:

$$Obj1 : \min \sum_{p \in P} c_p x_p \tag{1}$$

任务环成本c_p如下所示:

$$c_p = c_{\text{duty}} \sum_{i \in p} c_f \times ft_i + \sum_{d \in D_{\text{overnight}}} c_{\text{overnight}} + c_{\text{sit}}$$

其中, c_{duty}表示任务津贴, c_f是每小时的飞行津贴, ft_i为航班的飞行小时数, $c_{\text{overnight}}$为过夜成本, c_{sit}表示加机组成本.

第二目标是最大化机组利用率, 假设 I 是所有待执行的航班集合, i 是其中某个航班, $\forall i \in I$, 设ft_i为航班 i 的飞行时间, dt_i为航班 i 的执勤时间, $F_{p,t}$为任务环 p 在第 t 天的航班集合, 则第二目标函数$Obj2$如下:

$$Obj2 : \max \left(\sum_{i \in F_{p,t}} ft_i \Big/ \sum_{i \in F_{p,t}} dt_i \right) \tag{2}$$

假设a_{ip}是 0-1 变量,表示航班 i 是否在任务环 p 中, dep_i表示航班 i 的起飞机场, arr_i是航班 i 的降落机场, $tdep_i$是航班 i 的起飞时间, $tarr_i$是航班 i 的降落时间, $minct$ 表示最小间隔时间, $maxfts$ 表示机组每天的最大飞行时间, $maxdts$ 是机组每天的最大的执勤时间, j 航班是 i 航班的后续航班, 则要满足的基本约束条件有:

$$\sum_{p\in P} a_{ip}x_p = 1, \forall i \in I \tag{3}$$

$$dep_j = arr_i \tag{4}$$

$$tdep_j - tarr_i \geq minct \tag{5}$$

$$\sum_{i\in F_{p,t}} ft_i \leq fts \tag{6}$$

$$\sum_{i\in F_{p,t}} dt_i \leq maxdts \tag{7}$$

$$\begin{cases} date(tarr_i) - date(tdep_j) \leq maxdays \\ \forall i, j \in F_p, i \neq j, \forall p \in P \end{cases} \tag{8}$$

$$a_{ip}, x_p \in \{0,1\}, F_P, F_{p,t} \subseteq I \tag{9}$$

其中, 式 (3) 是航班覆盖约束, 即每个航班有且仅有一次出现在选中航班任务环中.

式 (4) 是航班地点衔接约束, 要求后一个航班的起飞机场要与前一个航班的降落机场一致.

式 (5) 是航班过站时间约束, 要求后一个航班的出发时间与前一个航班的降落时间之差要大于最小航班连接时间限制.

式 (6) 为飞行时间约束, 要求在每一天机组总飞行时间不能超过最大飞行时间限制.

式 (7) 为执勤时间约束, 要求在每一天机组总执勤时间不能超过最大执勤时间限制.

式 (8) 为航班任务环总跨度约束, 即每个任务环的日期跨度不能超过最长时间跨度约束, date 表示当前航班的日期.

式 (9) 中, a_{ip}, x_p是 0-1 变量, $F_P, F_{p,t}$是集合变量, 是所有航班集合 I 的子集.

3.2 航班任务环的生成

3.2.1 航班任务环生成算法

初始航班任务环生成阶段分成两步, 首先利用航班信息和相关约束通过改进的深度优先算法建立航班网络, 然后再利用 DFS 遍历所有航班网络生成初始航班任务环[22]. 航班网络的建立过程如算法 1 所示.

算法 1. 航班网络的建立

输入: 航班集合 I, 基地集合 B, 所有相关约束 C, 从机场出发的航班集合 $DepMap$

输出: 航班网络集合

```
1. 初始化 S_B, E_B   //航班网络虚拟节点, 分别表示从基地 B 出发和结束
2. for S_B 的每个子集 S_b do   //b 为具体的基地机场
3.     for i in DepMap[b] do   //DepMap[b] 表示从基地 b 出发的航班集合
4.         add childNode i to S_b
5.         for j in DepMap[i_arr] do
6.             if (i→j) 满足约束 C then
7.                 add childNode j to i
8.                 if j_arr=b;
9.                     add childNode E_b to j
10.                        return
11.                else
12.                    DFS(j,DepMap,C)
```

如算法 1 所示, 设虚拟起点为S_b, 对应的终点为E_b, 其中 b 是基地机场, $\forall b \in B$, B 为所有基地机场的集合, 寻找从起点S_b到终点E_b的航班路线, 最终生成 n 个航班网络, n 为基地机场的数量. 首先将所有航班按照起飞机场进行归类, 目的是在后续的航班衔接过程中满足地点衔接约束, 避免在建立航班网络过程中不必要的路径搜索, 提高搜索效率. 将每个航班都看作为网络中的一个节点, 节点包含航班的航班号、起飞时间和降落时间、起飞机场和降落机场等信息, 先建立从起点S_b到从基地 b 出发的航班节点的连接路径, 然后遍历从S_b到达航班节点, 对每个航班节点, 找到该航班的降落机场, 遍历从该机场出发的航班集合, 若满足后一航班的起飞时间和前一航班的降落时间差大于最小连接时间约束, 则建立从前一航班节点到后一航班节点的一条路径, 从当前节点进行递归, 直到某一航班节点的降落机场为 b, 则建立从该航班节点到终点E_b的路径. 航班网络的示意图如图 1 所示.

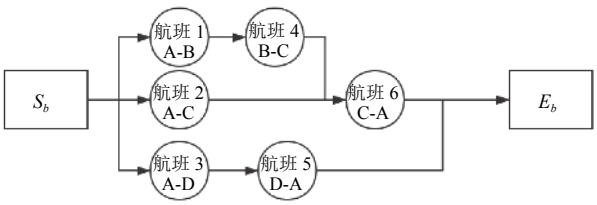

图 1 航班网络连接示意图

航班网络建立完成后, 初始航班环生成如算法 2 所示. 对于$\forall b \in B$, 利用深度优先算法遍历从S_b到

E_b的航班路径, 在遍历过程中统计路径中航班的飞行时间和执勤时间, 若到达某一航班节点时飞行时间或执勤时间已经超过了相关规定中对于飞行时间和执勤时间限制时, 不继续递归, 进行回溯, 回到上一个节点继续遍历, 直到满足约束的所有的路径都被遍历完, 则每一条可行路径都是一个合法的可行任务环, 此时初始航班任务环生成完成.

算法 2. 初始航班任务环生成

输入: 航班网络S_b, 约束 C
输出: 航班任务环 P

```
1.  P=∅  //初始化航班任务环集合
2.  for i in S_b.children do
3.      F_p=∅
4.      ft,dt=0    //初始化总飞行时间和总执勤时间
5.      add i to F_p
6.      if i.children not null and ft, dt 满足约束C：
7.          ft=ft+i_ft, dt=dt+i_dt
8.          for j in i.children do
9.              DFS(j,F_p)
10.     else
11.         add F_p to P
12.         F_p remove last
```

3.2.2 初始航班任务环生成实例

下面通过举例来说明航班任务环的生成, 给定一组航班计划表, 如表 1 所示, 其中包括 18 个航班, 每个航班包括航班号、执飞机型、起飞/降落机场, 起飞/到达时间等信息, 该航班计划中有 6 个机场, 其中广州为基地机场, 其余为非基地机场.

首先生成航班连接网络, 再根据航班网络进行深度优先搜索寻找满足约束的航班任务环. 假设该航班计划表的基本约束条件如下: 机组每天的总飞行时间不超过 8 小时, 机组每天总执勤时间不超过 14 小时, 航班衔接时间不少于 30 分钟, 任务环每天的航班数不超过 4 个. 则生成初始可行航班任务环的步骤如下;

(1) 按起飞机场对航班进行归类

该航班计划共有 6 个机场, 分别为广州、南昌、杭州、贵阳、桂林、长沙, 按起飞机场来划分可分为如下组合:

广州 (0, 5, 7, 9, 11, 16); 南昌 (1, 6, 8, 10); 贵阳 (2, 3, 17); 杭州 (4, 14); 桂林 (12); 长沙 (13, 15)

(2) 生成航班连接网络

由于任务环都是从基地出发, 且该航班计划只有广州一个基地机场, 因此建立虚拟起始节点到所有从广州出发航班节点的连接, 如图 2 所示.

表 1 某航班计划表

航班号	机型	起飞机场	降落机场	起飞时间	到达时间
0	B737	广州	南昌	2020/1/1 8:00	2020/1/1 9:00
1	B737	南昌	贵阳	2020/1/1 9:30	2020/1/1 10:30
2	B737	贵阳	杭州	2020/1/1 13:40	2020/1/1 14:40
3	B737	贵阳	广州	2020/1/1 16:40	2020/1/1 17:40
4	B737	杭州	广州	2020/1/1 17:10	2020/1/1 18:10
5	B737	广州	南昌	2020/1/1 10:20	2020/1/1 11:20
6	B737	南昌	杭州	2020/1/1 12:30	2020/1/1 13:30
7	B737	广州	贵阳	2020/1/1 15:10	2020/1/1 16:10
8	B737	南昌	广州	2020/1/1 18:00	2020/1/1 19:00
9	B737	广州	桂林	2020/1/1 12:30	2020/1/1 14:20
10	B737	南昌	长沙	2020/1/1 15:30	2020/1/1 17:00
11	B737	广州	贵阳	2020/1/1 15:20	2020/1/1 16:40
12	B737	桂林	长沙	2020/1/1 15:20	2020/1/1 18:50
13	B737	长沙	广州	2020/1/1 19:40	2020/1/1 21:30
14	B737	杭州	长沙	2020/1/1 14:40	2020/1/1 16:20
15	B737	长沙	广州	2020/1/1 17:10	2020/1/1 18:40
16	B737	广州	南昌	2020/1/1 11:40	2020/1/1 13:50
17	B737	贵阳	长沙	2020/1/1 17:20	2020/1/1 19:20

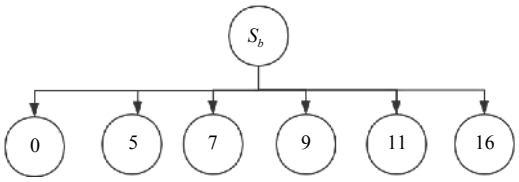

图 2 初始航班网络

然后在初始网络的基础上, 遍历每一个节点对每个节点, 寻找与其降落机场对应的航班集合, 如 0 号航班的降落机场为南昌, 则对起飞机场为南昌的集合进行遍历, 验证其是否满足航班衔接约束, 若不满足则舍弃, 若满足则建立连接, 将其加入航班网络中, 以此类推, 直到当前节点的到达机场为基地机场广州为止, 然后将终止节点与虚拟终止节点E_b, 航班网络建立完成. 由于涉及的路径较为复杂, 图 3 仅以从 0 号航班开始为例, 构建部分航班网络示意图.

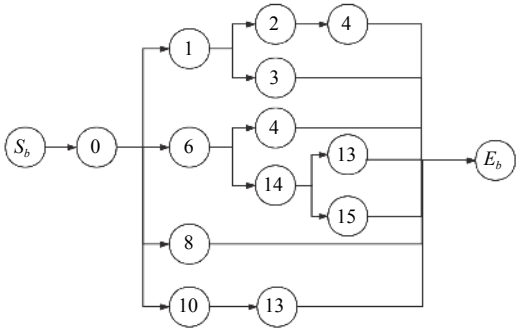

图 3 部分航班网络示意图

(3) 搜索可行航班任务环

根据上述生成的航班网络, 利用深度优先搜索, 加入约束条件, 如飞行时间、执勤时间等, 得到满足约束条件的航班任务环, 最终生成了 14 个合法的可行航班任务环, 生成的航班环如表 2 所示.

表 2 生成的合法航班环

航班环	P1	P2	P3	P4	P5	P6	P7
	0-8	0-1-3	0-6-4	0-1-2-4	0-6-14-15	5-8	5-6-4
飞行时间	120	180	180	240	310	120	180
执勤时间	720	640	670	670	700	580	530
机组利用率	0.167	0.281	0.269	0.358	0.442	0.207	0. 340
航班环	P8	P9	P10	P11	P12	P13	P14
	7-3	5-6-14-13	5-6-14-15	5-10-13	9-12-13	16-8	16-10-13
飞行时间	120	330	310	260	430	190	330
执勤时间	210	730	560	730	600	500	650
机组利用率	0.571	0.45	0.554	0.356	0.717	0.38	0.508

由表 2 可知, 生成的任务环的飞行时间和执勤时间均满足相应约束, 而并非所有航班环的机组利用率都是高效的, 机组利用率低导致机组在任务过程中空闲时间太长, 不能很好地体现机组的飞行价值, 而机组利用率也是优化求解的目标之一.

3.2.3 航空公司任务环生成实例

为了验证任务环生成算法的有效性, 本节采用国内某航空公司一周航班计划的真实数据 (对航班号进行了隐藏, 部分起飞和降落机场有所调整), 该航班计划共有 7 天, 共有 576 个航班, 每天的航班数量为 80 个左右, 主要基地机场为广州, 次基地有贵阳、三亚等, 根据航班网络生成算法建立航班连接网络, 由于页面限制, 图 4 仅展示第一天从广州基地出发的一个航班连接网络 (省略了首尾的虚拟节点).

图 4 中每个节点代表一个航班, 节点内为航班的虚拟编号, 为方便展示, 在节点左右两边分别标注了该航班的起飞机场和降落机场.

航班网络生成后, 利用航班环生成算法生成了高质量的航班任务环 2303 个, 部分航班环实例如表 3 所示, 其中日期表示任务环中跨越的日期, 时间单位 h 和 min 表示小时和分钟.

3.3 优化求解算法

初始航班任务环生成后, 利用 SMT 求解器 Z3 对模型进行表述, 假设输入航班集合为 I, 初始航班任务环集合为 IP, 定义 Z3 求解器模式为 Optimize, 定义不解释函数 $PC(p)$ 表示任务环 p 是否被选中, 函数的结果为 Bool 变量 (true 和 false), 则基于 Z3 的增量求解的过程如算法 3 所示.

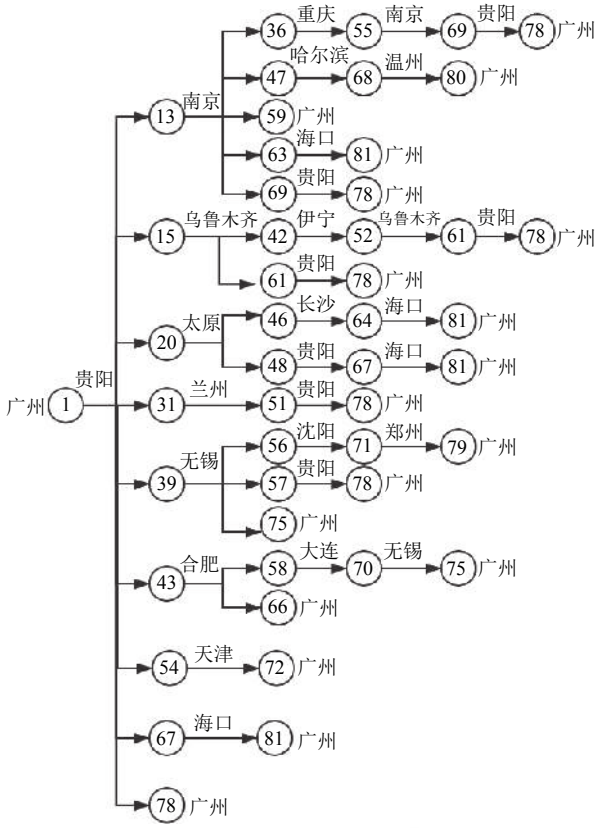

图 4 航班连接网络图

表 3 航班任务环实例

编号	日期	任务环	飞行时间	值勤时间	机组利用率
1	周一 周二	广州-南昌-兰州-南昌 南昌-兰州-南昌-广州	13 h 10 min	19 h 5 min	0.690
2	周一 周二	广州-贵阳-南京-重庆-南京 南京-长春-南通-广州	16 h 45 min	30 h 15 min	0.554
3	周一 周二	广州-郑州-沈阳-无锡-贵阳 贵阳-温州-贵阳-合肥-广州	20 h	36 h 40 min	0.652
4	周一 周二 周三	广州-贵阳-兰州 兰州-南昌 南昌-兰州-南昌	12 h 40 min	22 h 30 min	0.563
5	周三 周四	广州-合肥-包头-合肥 合肥-营口-合肥-广州	13 h 30 min	19 h 20 min	0.698
6	周四 周五	贵阳-合肥-大连-无锡-广州 广州-南昌-兰州-贵阳	14 h 55 min	29 h 10 min	0.511
7	周五 周六 周日	广州-南京-长春-南京-海口 海口-贵阳-合肥 合肥-营口-合肥-广州	21 h 35 min	33 h 45 min	0.640
8	周六 周日	三亚-贵阳-天津 天津-贵阳-三亚	10 h 40 min	16 h 10 min	0.665

算法 3. 基于 Z3 的增量优化求解

输入: 机场的集合 *AP*, 基地集合 *BS*, 航班的集合 *I*(1.. *n*), 任务环生成函数 *GenPairing*(), 参数 *level*
输出: 最终任务环 *FP*, 成本 *costs*

```
1.  P=∅, costs=0, fts, dts=0, S=Optimize()
2.  for (l=0; l≤level; l++)
3.    P*, Cost*=GenPairing(l)    //根据level生成航班任务环
4.    Cst=∅
5.    fori in I.length do
6.      P1,P2,⋯,Pk⊆P*∨P 且包含航班i
7.      S.add_constain(Ci,l→PC(P1)∨PC(P2)∨⋯∨PC(Pk))
8.      Cst.add(Ci,1∧Ci,2∧⋯∧Ci,l)
9.      for p in P*∨P, Pj≠Pi 且 Pi∧Pj≠∅ do
10.       S.add_constain(¬PC(Pi)∨¬PC(Pj))
11.     for each p in P*, each cost in Cost* do
12.       costs add IF(PC(p),cost,0)
13.       fts add IF(PC(p),ft,0)
14.       dts add IF(PC(p),dt,0)
15.   ut=fts/dts
16.   S.minimize(costs), S.maxmize(ut)
17.   S.push(Cst)  //使约束生效
18.   Wait_for(costs,ut=Check(S))
19.   if unsat:
20.     S.pop()
21.     S.add_constrain(¬Cst)  //使约束失效
22.     P=P∨P*
23.   else if TimeOut
24.     return –1
25.   else
26.     return costs,ut,PC
```

如算法 3 所示, 输入的信息包括航班信息 *I*, 航班任务环生成函数 *GenPairing*(), 其中 *GenPairing*() 可以输入的参数 *level* 生成不同层次的任务环, 如 *level*=0 时生成当天返回基地的任务环, *level*=1 时生成当天未返回基地的任务环等.

算法遍历 *level* 依次生成不同层次的任务环, 5–7 行是航班覆盖约束, 要求每个航班至少被 1 个任务环覆盖, $C_{i,1}$为控制约束变量, 决定覆盖约束是否生效. 9–10 行是唯一性约束, 即任意两个任务环P_i和P_j不能同时包含同一个航班. 11–14 行利用 Z3 的 IF 函数, 将满足 *PC*(*p*)=true 的任务环 *p* 的飞行时间 *ft*、执勤时间 *dt*、成本 *cost* 增加到总时间 *fts* 和 *dts* 和成本 *costs* 中. 17–26 行利用 Z3 的增量求解技术,先将控制变量 *Cst* 对应的约束生效, 利用求解器 *S* 进行 Check() 求解, 若超过规定时间则返回–1, 若求解器 *S* 返回结果为 *unsat* 时, 则进行回溯, 使控制的约束失效, 更新任务环 *P*, 并进入下一轮循环; 当求解器 *S* 在给定的时间内给出 *sat* 结果时, 返回优化求解后的最终任务环以及总成本 *costs* 和机组利用率 *ut*.

下面仍以表 1 的航班计划表为例, 阐述优化求解的过程. 航班总数为 18, 经过 *GenPairing*(0) 生成了第一阶段当天返回基地的初始航班任务环数量为 16, 然后对其增加相应的约束进行 *Check*() 检查, 发现结果为 *unsat*, 使部分约束失效后进行下一阶段 *GenPairing*(1) 生成当天未返回基地的不完全任务环 11-17、0-10 等, 由于未返回基地增加过夜成本, 加入先前的任务环中, 添加相应约束进行求解, 仍为 *unsat*, 再下一阶段 *GenPairing*(2) 增加覆盖次数较少的航班 2、7、10 等加入任务环后, 结果为 *sat*, 最终生成的任务环如表 4 所示.

表 4 优化求解后的任务环

任务环	0-1-2-4	5-6-14-15	7-3	9-12-13	16-8	11-17	10	总计
成本	1800	2200	1400	2600	1800	1800	1200	12 800
机组利用率	0.358	0.554	0.571	0.717	0.38	0.754	0.75	0.62

由最后的结果可知, 生成的任务环覆盖了所有的航班, 且每个航班有且仅被覆盖一次, 同时, 总成本也较低, 机组利用率超过了 0. 6, 说明该方法是有效的.

4 实验分析

4.1 实验环境及实验分析

本文对航空公司机组排班问题进行优化求解, 实验环境选择个人电脑, CPU 为 AMD Ryzen5 1600X 6 核心 12 线程, 内存 16 GB, Win10 系统, 编程语言选择 Python, SMT 求解器选择 Python 版本的 Z3Py[23], 实验数据来自 Kasirzadeh 等[24] 提供的航空公司的一组航班计划数据. 本文选取前 7 天航班计划数据来验证求解结果和性能, 按照跨越天数将其分为 7 组, 数据的基本信息如表 5 所示, 部分航班时刻表如表 6 所示.

表 6 中, 表头 leg_nb 表示航班编号, airport_dep 与 airport_arr 分别表示航班的出发机场和到达机场, date_dep、hour_dep 表示航班出发的日期和时间, date_arr、hour_arr 表示航班到达日期和到达时间.表中, BASE*表示该机场为基地机场, AIR*表示机场为非基地机场. 经过改进的航班网络生成和 DFS 算法, 有效的减少了生成的初始航班任务环的数量,按照不同的 *level* 参数, 生成用于求解的航班环数量如表 7 所示.

优化求解阶段, 利用 Z3 的增量求解机制, 依次对不同 *level* 生成的任务环进行 check 检查其是否满足相应约束, 直到结果为 *sat*, 进行优化求解. 实验表明, 当

level=2 时各组数据生成的任务环均可满足相应的约束.

假设机组每天最大飞行时间为 8 小时, 最大执勤时间为 14 小时, 最小衔接时间为 30 分钟, 任务津贴为 1000, 每小时飞行津贴为 400, 过夜成本为 500, 加机组成本为 500, 机组在任务开始前 45 分钟报到, 任务结束后 30 分钟完成执勤, 优化求解后生成最终的航班任务环及机组利用率等结果如表 8 所示.

表 5 航班数据基本信息

天数	基地	机场数	航班数
1	BASE1/BASE2/BASE3	15	28
2	BASE1/BASE2/BASE3	16	64
3	BASE1/BASE2/BASE3	16	100
4	BASE1/BASE2/BASE3	16	136
5	BASE1/BASE2/BASE3	17	167
6	BASE1/BASE2/BASE3	19	199
7	BASE1/BASE2/BASE3	19	234

表 6 部分航班时刻表

leg_nb	airport_dep	date_dep	hour_dep	airport_arr	date_arr	hour_arr
LEG_01_0	BASE1	2000-01-01	12:00	AIR1	2000-01-01	13:13
LEG_01_1	AIR1	2000-01-01	14:05	BASE2	2000-01-01	15:19
LEG_01_2	AIR3	2000-01-01	17:45	BASE2	2000-01-01	18:27
LEG_01_3	BASE2	2000-01-01	19:24	AIR2	2000-01-01	20:15
LEG_01_4	AIR5	2000-01-01	20:35	BASE2	2000-01-01	21:49
LEG_01_5	BASE2	2000-01-01	23:11	AIR4	2000-01-02	00:22

表 7 各组航班环数量

组别	1	2	3	4	5	6	7
航班数	28	64	100	136	167	199	234
level=0	10	42	102	163	217	266	327
level=1	37	85	161	238	302	373	462
level=2	65	149	262	374	469	572	696

表 8 模型求解结果

组别	最终任务环数量	总飞行时间(min)	总执勤时间(min)	机组利用率	总成本	求解时间(s)
1	11	3034	4939	0.61	26 700	0.39
2	29	7193	12 400	0.58	59 100	0.89
3	38	11 352	20 387	0.56	77 800	2.18
4	55	15 511	26 162	0.59	105 800	4.16
5	61	19 324	34 823	0.55	110 200	7.3
6	75	23 045	41 411	0.56	133 600	35.1
7	86	27 017	48 515	0.56	174 600	405.3

从求解结果可知, 该算法能有效的从给定的航班计划中选择符合各项约束的任务环. 经过验证, 生成的任务环能覆盖所有航班, 且机组利用率均超过了 55%, 有效的减少了机组的空闲等待时间, 有效的缩短了机组执勤时间, 同时最小化的成本可为航空公司减少一定的机组成本支出, 增加航空公司的收益.

4.2 实验对比

为了验证实验算法的性能, 本文与 Agustín 等[25]提出的基于偏向随机化的多起点元启发式算法以及人工编排方案进行比较分析, 实验数据来自某小型航空公司的真实案例, 数据来源于文献 [25]. 共有 41 个航班, 跨越 5 天, 航班的最小连接时间为 45 min, 只有一个基地机场 Madrid (MAD).

经过实验, 得出实验结果如表 9 所示.

表 9 实验结果对比

参数	人工	M-BR	SMT
任务环数	6	5	5
飞行时间	69 h 40 min	67 h 40 min	66 h 40 min
值勤时间	120 h 40 min	125 h 35 min	114 h 10 min
机组利用率	0.577	0.539	0.593
机组成本	46 500	43 700	43 300
CPUt (s)	—	20	0.36

从实验结果可知, SMT、M-BR 与人工编排相比生成的任务环数量更少, 飞行时间更短, 机组成本更低, 这是由于人工编排产生了许多不必要的加机组航班, 增加了飞行时间和机组成本; SMT 相比于 M-BR, 机组的利用率更高且求解时间更短.

通过分析实验结果, 本文提出的 SMT 算法相较于人工和 M-BR 算法, 求解效率更高, 总值勤时间更少, 机组利用率更高, 更符合航空公司机组排班阶段对任务环质量的要求, 可以有效的减少机组成本和提高机组利用率.

5 总结

本文采用基于 SMT 的研究方法, 对航空公司运营计划中机组排班计划问题的机组排班问题进行了研究, 综合考虑了民航管理局等制定的关于飞行安全、人员要求等多种约束, 建立了基于 SMT 的机组排班优化模型, 采用改进的深度优先搜索算法生成初始航班任务环, 并利用 SMT 求解器 Z3 进行模型的优化求解, 并使用增量求解. 实验表明, 改进的 DFS 算法能更快速的生成更少更优质的初始航班任务环, 基于 SMT 的 Z3 求解器可以更好的表述约束和求解约束, 并可以增量

求解, 最终生成机组利用率高且质量较好的航班任务环, 帮助航空公司制定更好的排班计划以及节约机组成本, 增加收益.

在之后的研究中, 可对机组排班的下一个阶段机组人员分配问题进行研究, 制定完整的机组排班计划, 以及在考虑延误等突发情况下如何增加机组排班的鲁棒性进行进一步的研究, 更贴近实际, 减少航班延误造成的损失.

参考文献

1 CCAR-121 R6 大型飞机公共航空运输承运人运行合格审定规则. https://www.woair.com/zhengfu/news/2020-06-09/19.html. (2020-06-09).

2 Aggarwal D, Saxena DK, Emmerich M, *et al*. On large-scale airline crew pairing generation. 2018 IEEE Symposium Series on Computational Intelligence (SSCI). Bangalore: IEEE, 2018. 593–600. [doi: 10.1109/SSCI.2018.8628699]

3 McCarver M. Airline Crew Scheduling Problem. 2019 NCUR, 2019.

4 Haouari M, Mansour FZ, Sherali HD. A new compact formulation for the daily crew pairing problem. Transportation Science, 2019, 53(3): 811–828. [doi: 10.1287/trsc.2018.0860]

5 Zhang ZZ, Zhou MC, Wang JH. Construction-based optimization approaches to airline crew rostering problem. IEEE Transactions on Automation Science and Engineering, 2020, 17(3): 1399–1409. [doi: 10.1109/TASE.2019.2955988]

6 Kornilakis H, Stamatopoulos P. Crew pairing optimization with genetic algorithms. In: Vlahavas IP, Spyropoulos CD, eds. Methods and Applications of Artificial Intelligence. Berlin: Springer, 2002. 109–120.

7 赵天洋. 航班机组自动排班系统研究 [硕士学位论文]. 沈阳: 沈阳理工大学, 2016.

8 吴苏阳. 航空公司飞行任务的自动组环优化算法研究与实现 [硕士学位论文]. 上海: 复旦大学, 2016.

9 Aggarwal D, Saxena DK, Bäck T, *et al*. On initializing airline crew pairing optimization for large-scale complex flight networks. arXiv: 2003.06423, 2020.

10 张文成, 熊静, 张虹, 等. 基于改进二进制粒子群算法的机组配对优化. 上海工程技术大学学报, 2020, 34(1): 34–40. [doi: 10.3969/j.issn.1009-444X.2020.01.006]

11 Chen CH, Chou FI, Chou JH. Multiobjective evolutionary scheduling and rescheduling of integrated aircraft routing and crew pairing problems. IEEE Access, 2020, 8: 35018–35030. [doi: 10.1109/ACCESS.2020.2974245]

12 Marsten RE, Shepardson F. Exact solution of crew scheduling problems using the set partitioning model: Recent successful applications. Networks, 1981, 11(2): 165–177. [doi: 10.1002/net.3230110208]

13 Graves GW, McBride RD, Gershkoff I, *et al*. Flight crew scheduling. Management Science, 1993, 39(6): 736–745. [doi: 10.1287/mnsc.39.6.736]

14 蓝伯雄, 张米. 机组排班的混合集合规划方法研究. 运筹与管理, 2014, 23(2): 175–182. [doi: 10.3969/j.issn.1007-3221.2014.02.024]

15 Zeighami V, Saddoune M, Soumis F. Alternating Lagrangian decomposition for integrated airline crew scheduling problem. European Journal of Operational Research, 2020, 287(1): 211–224. [doi: 10.1016/j.ejor.2020.05.005]

16 金继伟, 马菲菲, 张健. SMT 求解技术简述. 计算机科学与探索, 2015, 9(7): 769–780. [doi: 10.3778/j.issn.1673-9418.1405041]

17 Sebastiani R, Tomasi S. Optimization in SMT with *LA*(Q) cost functions. Proceedings of the 6th International Joint Conference on Automated Reasoning. Manchester: Springer, 2012. 484–498.

18 Klein G. Operating system verification-an overview. Sadhana, 2009, 34(1): 27–69. [doi: 10.1007/s12046-009-0002-4]

19 Blackham B, Liffiton M, Heiser G. Trickle: Automated infeasible path detection using all minimal unsatisfiable subsets. 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS). Berlin: IEEE, 2014. 169–178. [doi: 10.1109/RTAS.2014.6926000]

20 De Moura L, Bjørner N. Z3: An efficient SMT solver. Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2008. 337–340.

21 The SMT-LIBv2 language and tools: A tutorial. http://smtlib.github.io/jSMTLIB/SMTLIBTutorial.pdf. (2013-11-23).

22 鲁红珍. 航空公司机组航班任务串优化方法研究 [硕士学位论文]. 广汉: 中国民用航空飞行学院, 2012.

23 The Z3 theorem prover. https://github.com/Z3Prover/z3. [2020-12-10].

24 Kasirzadeh A, Saddoune M, Soumis F. Airline crew scheduling: Models, algorithms, and data sets. EURO Journal on Transportation and Logistics, 2017, 6(2): 111–137. [doi: 10.1007/s13676-015-0080-x]

25 Agustín A, Gruler A, De Armas J, *et al*. Optimizing airline crew scheduling using biased randomization: A case study. Proceedings of the 17th Conference of the Spanish Association for Artificial Intelligence. Salamanca: Springer, 2016. 331–340.