中国科学: 信息科学 2016 年 第 46 卷 第 7 期: 834–854
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
论 文

求解多阶段护士排班问题的带权禁忌搜索算法

苏宙行, 王卓*, 吕志鹏

华中科技大学计算机科学与技术学院智慧计算与优化实验室, 武汉 430074
* 通信作者. E-mail: wang_zhuo@hust.edu.cn

收稿日期: 2016–03–08; 接受日期: 2016–04–14
国家自然科学基金 (批准号: 61370183, 61100144) 和 2013 教育部新世纪优秀人才支持计划资助项目

摘要 本文研究了多阶段护士排班问题, 该问题由第 2 届国际护士排班竞赛提出, 在医疗优化领域具有重要的意义. 针对多阶段护士排班问题, 本文提出了一种带权禁忌搜索算法. 该算法使用了 3 种互斥的简单邻域结构与一种复合邻域结构, 并根据其适应性动态调整搜索各邻域的概率. 同时, 通过调整各护士的惩罚权重来实现搜索过程中集中性与疏散性的平衡. 为了应对各个独立的阶段缺乏全局信息的问题, 提出了一种对全局约束的近似评估策略. 算法还针对自适应的邻域选择策略, 以减少多种邻域的整体计算开销为目标设计了邻域评估的缓存策略, 进一步提升了算法的执行效率. 算法在竞赛使用的 60 个算例上的计算结果表明了算法的有效性, 最终在第 2 届国际护士排班竞赛决赛中排名全球第 4. 此外, 本文对算法中的关键要素进行了对比分析, 表明了这些组成部分的方案选择和参数设置的合理性.

关键词 护士排班 禁忌搜索 时刻表规划 人员排班 元启发式 组合优化

1 引言

随着医疗行业的发展, 医院拥有了越来越多的医务人员、医疗设备和病房, 同时也需要满足更多病人的需求. 于是, 如何更合理地利用现有资源来为更多的病人提供更好的服务, 其重要性正日益凸显. 在众多资源当中, 医护人员起到了支配性作用, 无疑是优化资源利用率的关键因素. 所以, 合理的人员排班显得尤为重要, 既可以保证病人的病情不被延误, 又可以保证医务人员有足够的休息时间, 以保持较高的工作效率.

护士排班是一个典型的人员排班问题, 需要给出每天每个班次有哪些护士需要上班, 即一个人员排班表. 该排班表必须满足多种硬约束, 如各班次各岗位的人数达到需求下限、避免不合理的排班模式以及一个护士在同一天只能排一个班次等. 同时, 排班表还应尽量满足工作量上限、连续工作天数上下限、周末工作天数上限以及护士的特殊需求等软约束 [1~3].

由于每天每班次每岗位上班的护士可以从拥有该岗位所需技能的护士中任意挑选, 其人员选择已经具有组合数级别的复杂度, 同时对整个排班周期进行排班又需要指数级别的复杂度来完成. 此外,

引用格式: 苏宙行, 王卓, 吕志鹏. 求解多阶段护士排班问题的带权禁忌搜索算法. 中国科学: 信息科学, 2016, 46: 834–854, doi: 10.1360/N112015-00284

© 2016《中国科学》杂志社

众多的软约束之间相互作用, 此消彼涨. 因此, 护士排班问题无论是对医院的管理人员还是学者来说, 都是一个极其复杂的问题, 对其进行求解优化是一项艰巨的挑战.

护士排班问题在近几十年来得到了广泛的研究, 而最近十年研究尤为深入. 在众多求解算法中, 主要有精确算法和启发式算法两大类别. 精确算法有 Isken [4] 和 Glass 等 [5] 使用的整数规划方法, 以及 Beliën 等 [6] 提出的分支定价方法. 此外, He 等 [7] 使用列生成算法对护士排班问题进行了建模与求解.

另一方面, 启发式算法也得到了众多研究者的关注. Burke 等 [8] 提出了针对护士排班问题的混合禁忌搜索, 之后又相继提出了模因算法 [9]、变邻域搜索 [10,11]、分散搜索 [12] 和变深度搜索 [13] 来求解护士排班问题. 此外, 针对护士排班问题还有众多解决方案, 例如 Bai 等 [14] 提出的混合进化算法、Anwar 等 [15] 提出的超启发式算法以及 Huang 等 [16] 提出的进化算法.

由于护士排班问题是从实际应用中提炼出来的问题, 不同医院的不同需求导致该问题长期以来没有统一的问题定义与算例, 因此难以比较各种算法的优劣. 于是, 由鲁汶大学的 CODeS 课题组主办的护士排班竞赛 [17,18] 应运而生. 在第 1 届国际护士排班竞赛1)中, Valouxis 等 [19] 提出的整数规划与局部搜索相结合的混合算法、Burke 等 [20] 提出的分支定价算法以及 Lü 等 [21] 提出的自适应邻域搜索算法分别证明了其有效性.

为了更加贴近医院排班的实际场景, 第 2 届国际护士排班竞赛2)正式提出了多阶段护士排班问题, 在提炼出常见的实际约束的基础上, 引入了多阶段的概念以适应整个排班周期中需求变化的特点, 从而使得排班更具灵活性, 与此同时也增加了求解的挑战性. 此次竞赛全球共有 15 支队伍参加, 最终 7 支队伍进入决赛, 本文提出的算法在决赛中排名第 4.

由于医院对医护人员的需求量具有不确定性, 周期较长的排班很容易因工作量的变化而需要重新调整, 因此第 2 届国际护士排班竞赛除了保留了传统的约束之外, 还将整个排班周期划分成多个较短的阶段, 对当前阶段进行排班时无法得知后续阶段的人员需求, 同时无法更改先前阶段的排班计划. 在这种情况下, 仍然需要考虑平衡所有护士在整个排班周期中的工作量.

本文提出的算法为针对第 2 届护士排班竞赛而设计的多邻域带权禁忌搜索. 首先, 该算法在计算目标函数值时给每个护士设置了不同的权重, 该权重根据当前解的改进潜力动态调整. 在当前解还有改进空间时, 算法将加强集中性, 将所有护士设置成相同的权重, 对原始目标函数进行优化; 在一定迭代次数没有改进最优解时, 算法将增强疏散性, 调整权重对软约束违反较多的护士进行针对性优化. 其次, 算法使用了多种结构和复杂度均不相同的邻域动作, 并根据邻域动作的效果动态调整探索各邻域的概率. 此外, 算法对复杂邻域动作的增量评估结果进行了缓存, 有效减少了重复计算. 最后, 算法设计了辅助目标函数与全局约束的估算策略, 以实现对跨阶段约束的优化.

本文接下来的部分将按如下结构组织. 第 2 节对第 2 届护士排班竞赛的问题模型进行了简要的介绍. 第 3 节对带权禁忌搜索算法进行了详细的描述和分析. 第 4 节展示了算法的计算结果和竞赛结果. 第 5 节通过对比实验对算法的重要策略进行了深入的分析和讨论. 最后一节对全文进行总结.

2 问题定义

本文讨论的护士排班问题的优化目标是在满足人员需求的情况下尽可能地平衡护士的工作强度, 以提升医院的效益. 每一天有若干时间段不同的班次, 如早班和夜班. 一个护士拥有若干技能, 如某一

1) 官方网站为 http://www.kuleuven-kulak.be/nrpcompetition.

2) 官方网站为 http://mobiz.vives.be/inrc2/.

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	h_3, k_2		h_1, k_1	h_1, k_2	h_1, k_1	h_2, k_2	h_2, k_2
n_2	h_2, k_1	h_2, k_2			h_2, k_2	h_2, k_1	h_2, k_1
n_3	h_3, k_1	h_3, k_2	h_3, k_1	h_3, k_1			
n_4				h_3, k_2	h_3, k_2	h_3, k_2	h_3, k_2
n_5	h_1, k_2	h_1, k_2	h_2, k_2	h_2, k_2		h_1, k_2	h_1, k_2

图 1　解向量示例

Figure 1　Example of solution

个护士掌握了护士长和普通护士两个技能. 为了表述方便, 将一天中任意一个班次与任意一个技能的组合称作一个时间槽. 例如, 如果有早班和晚班两个班次, 以及护士长和普通护士两个技能, 那么每天就有 (早班, 护士长)、(晚班, 护士长)、(早班, 普通护士) 和 (晚班, 普通护士) 一共 4 个时间槽. 该问题将给出固定数量的护士、排班周期的长度以及每天每个时间槽的人数需求, 要求得出一个排班计划, 确定每天每个时间槽安排哪些护士上班, 即确定是否安排一个护士在某一天上班, 如上班, 该护士使用其掌握的哪个技能上哪一个班. 此外, 整个排班周期被划分为多个阶段, 依次对各阶段进行排班. 在对某一阶段进行排班时, 无法修改先前阶段的排班, 也无法获知后续阶段每天的时间槽对护士人数的需求信息, 只有特殊的边界信息 (历史) 能从前一个阶段传递到后一阶段. 该排班计划必须满足若干硬约束, 同时尽可能减少对各项软约束的违反. 每个护士有一份工作合同, 合同详细规定了工作强度的最佳范围. 工作强度包括总工作天数、周末工作天数、连续工作天数、连续相同班次数以及连续休息天数等指标. 此外, 每个护士可以请求在指定的日期不为其排班.

整个周期由 t 个长度相等的连续且不重叠的阶段组成. 令 D 为排班周期中各天组成的集合, D_i 为阶段 i 中各天的集合, $D = D_1 \cup D_2 \cup \cdots \cup D_t$; N 为护士的集合; H 为班次的集合; K 为技能的集合; X_i^H 为一个 $|N| \times |D_i|$ 的矩阵, 用于表示各护士在阶段 i 中各天被安排至哪个班次, 其中, 休息 (ShiftOff) 可以看作一个特殊的班次; X_i^K 为一个 $|N| \times |D_i|$ 的矩阵, 用于表示各护士在阶段 i 中各天使用哪个技能上安排的班次; X_i^H 和 X_i^K 共同构成了阶段 i 的解向量 X_i, 整个排班周期的解向量为 X, $X = [X_1, X_2, \ldots, X_t]$. 图 1 给出了一个 5 个护士 7 天的解向量示例. 从图中可知每个护士每一天的工作安排, 如护士 n_1 在第 d_1 天被安排在时间槽 (h_3, k_2), 护士 n_2 在第 d_3 天休息.

该问题一共有 4 个必须满足的硬约束:

- **H_1. 单一指派约束.** 每个护士每天最多只能被安排到一个时间槽. 在本文提出的解向量的表示方式下, 该硬约束将自动满足.
- **H_2. 人数下限约束.** 每天每个时间槽的排班人数不能小于其人数需求的最小值.
- **H_3. 后继约束.** 同一个护士连续两天的排班不能出现某些特定的模式. 例如, 一个护士在某天上了夜班, 则第二天不能再安排上早班.
- **H_4. 技能约束.** 安排到某个时间槽的护士必须具备对应的技能. 例如, 安排在 (早班, 护士长) 的护士必须具备护士长这个技能.

此外, 还有 8 个应尽量避免违反的软约束:

- **S_1. 最优人数约束.** 每天每个时间槽安排的护士数量不得小于给定的最优值, 每缺少一个护士都受到 1 个单位的惩罚. 该最优值大于等于 H_2 规定的相应时间槽的人数需求最小值.

• **S_2. 连续工作日约束.** 每个护士的连续工作天数必须在给定的区间内, 每个超出或不足的工作日都受到 1 个单位的惩罚.

• **S_3. 连续相同班次约束.** 每个护士被连续安排到同一班次的天数必须在给定的区间内, 每个超出或不足的工作日都受到 1 个单位的惩罚.

• **S_4. 连续休息约束.** 每个护士的连续休息天数必须在给定的区间内, 每个超出或不足的休息日都受到 1 个单位的惩罚.

• **S_5. 工作偏好约束.** 如果一个护士请求不在某天某班次上班, 但仍然被安排在该天该班次上班, 将受到 1 个单位的惩罚.

• **S_6. 完整周末约束.** 如果一个护士的合同规定其应有完整的周末, 则其在周末的排班必须是两天均上班或者两天均休息, 否则将受到 1 个单位的惩罚.

• **S_7. 总排班数约束.** 每个护士在整个排班周期中的总工作天数必须在给定的区间内, 每个超出或不足的工作日都要受到 1 个单位的惩罚.

• **S_8. 总周末排班数约束.** 每个护士在整个排班周期中被排班了的周末总数不得大于给定的最大值, 每超出一个工作周末都要受到 1 个单位的惩罚. 其中周六或周日任意一天被排班则周末算作工作, 否则为休息.

其中所有硬约束可以在各阶段中单独考虑, 各阶段满足即整个周期的排班合法; 软约束中除 S_7 与 S_8 外其他约束均可在各阶段中考虑, 各阶段惩罚的累加和即为整个周期的相应惩罚. S_7 与 S_8 是横跨整个周期的约束, 由于整个周期的排班是分阶段进行的, 因此只有在排最后一个阶段时才能准确计算这两个软约束的违反程度. 对于其他阶段, 可以采取近似评估策略考虑, 以防止前期由于不考虑而违反过多. 另外, 除了 H_2 与 S_1 是与时间槽相关的约束, 其他约束只针对单个护士, 护士之间不会相互影响. 问题的求解目标为, 在整个排班周期结束时, 解向量 X 满足硬约束 $H_1 \sim H_4$, 同时使违反软约束 $S_1 \sim S_8$ 受到的惩罚最小. 假设软约束 S_i 在所有阶段的惩罚之和为 $f_i(X)$, 权重系数为 c_i, 则合法解 X 的目标函数 $f(X)$ 如式 (1) 所示,

$$f(X) = \sum_{i=1}^{8} c_i \times f_i(X), \tag{1}$$

该问题的目标即为找到一个满足所有硬约束同时目标函数值最小的解.

已知数据主要包括场景、需求和历史 3 类.

场景信息为应用于所有阶段的公共数据, 其规定了排班周期的长度, 可能出现的班次和技能, H_3 中涉及的非法后继班次的模式, 不同种类的合同的内容, 以及可供排班的护士与他们签订的合同类型. 其中合同规定了 S_2 和 S_4 中涉及的连续天数的上下限、S_6 中涉及的是否要求有完整周末、S_7 中涉及的总排班数区间以及 S_8 中涉及的总周末排班数上限.

需求信息仅作用于单个阶段, 其中包括用于判断 H_2 和 S_1 的违反程度的人数需求, 即一阶段中每天每个时间槽的最低和最优护士数量, 以及 S_5 中描述的护士请假要求.

历史信息即从前一阶段传递给后一阶段的数据, 用于评估跨阶段约束的违反情况. 典型的历史信息包含各护士的累计排班数、累计周末排班数、上阶段最后一天的排班、上阶段最后的连续排班天数与连续休息天数等. 求解器的实现者也可以将任意可获取的数据以自定义格式保存供后续阶段使用.

具体数据格式以及各约束在边界情况下的处理请参考竞赛的问题描述文档 [18].

3 带权禁忌搜索

3.1 算法流程概览

护士排班问题的求解过程将按阶段依次执行, 直到覆盖整个排班周期. 如算法 1 所示, 每个阶段以场景信息 Scenario、上阶段的历史信息 h_i 与本阶段的需求信息 Weekdata_i 为输入, 调用带权禁忌搜索算法, 输出本阶段的解向量 X_i 与供下阶段使用的历史信息 h_{i+1}. 在本次竞赛中, 排班周期以周为单位划分阶段, 通常为 4 周或 8 周, 且参赛者只需设计实现一周排班的求解算法, 由竞赛组织者重复调用该算法来完成整个周期所有阶段的排班.

算法 1 护士排班问题求解过程

Input: 算例数据 Scenario, InitialHistory, Weekdata;
Output: 求得的最优排班计划 X;
  $h_0 \Leftarrow$ InitialHistory;
  $i \Leftarrow 0$;
  while $i < t$ **do**
    $X_i \Leftarrow \text{WeightedTabuSearch}(\text{Scenario}, h_i, \text{Weekdata}_i)$; // 见算法 2
    $h_{i+1} \Leftarrow \text{GenerateHistory}(\text{Scenario}, h_i, X_i)$;
    $i \Leftarrow i + 1$;
  end while

对单阶段进行排班使用了带权禁忌搜索算法, 其伪代码如算法 2 所示. 算法首先产生一个合法的初始解作为当前解, 然后交替使用集中性和疏散性的多邻域禁忌搜索, 每轮先使用集中性搜索对当前解进行改进, 更新全局最优解 X_i 之后, 再使用疏散性搜索进行扰动. 到达给定的运行时间后算法终止并输出找到的最优排班. 由于只是针对一个阶段进行排班, 需要定义单个阶段解 X_i 的目标函数

$$f^{\mathrm{S}}(X_i) = \sum_{j=1}^{6} c_j \times f_j^{\mathrm{S}}(X_i) + \sum_{j=7}^{8} c_j \times g_j^{\mathrm{S}}(X_i), \tag{2}$$

其中 $f_j^{\mathrm{S}}(X_i)$ 为软约束 S_j 在阶段 i 的惩罚之和, 其计算方式和针对全周期解向量 X 的计算方式 f_j 相同. g_7^{S} 和 g_8^{S} 是针对 S_7 和 S_8 的近似评估函数.

接下来将详细介绍用于单个阶段求解的算法实现细节.

算法 2 求解护士排班问题的带权禁忌搜索算法框架

Input: 算例数据 Scenario, History, Weekdata;
Output: 求得的本阶段的最优排班计划 X_i;
  $X_i \Leftarrow$ GenerateInitialSolution(); // 生成初始解, 见 3.2 小节
  $s \Leftarrow X_i$; // s 表示当前解
  repeat
    ResetWeight(w); // 见 3.5 小节
    $s \Leftarrow \text{TabuSearch}(s, w)$; // 集中性多邻域禁忌搜索, 见 3.4 小节算法 3
    UpdateOptima(s, X_i); // 更新全局最优解 X_i
    $s \Leftarrow \text{SelectRandomly}(s, X_i)$; // 从 s 和 X_i 中等概率随机选择一个进行之后的疏散性搜索
    $w \Leftarrow \text{AdjustWeightToBiasNurseWithGreaterPenalty}(s)$; // 调整惩罚权重, 见 3.5 小节
    $s \Leftarrow \text{TabuSearch}(s, w)$; // 疏散性多邻域禁忌搜索, 见 3.4 小节算法 3
  until 满足停止条件

3.2 初始解生成

初始解的生成使用了贪心构造的方法, 从阶段的第一天到最后一天依次填充时间槽以确定排班计划. 对于每一天 d, 统计每个技能 k 对护士的总最低需求量 $r_{d,k}$, 即当天技能为 k 的各时间槽的人数下限之和, $r_{d,k}=\sum_{h\in H}\text{MinDemand}(d,h,k)$, 其中 MinDemand 函数表示第 d 天班次 h 技能 k 的时间槽 (h,k) 的人数下限 (硬约束 H_2). 然后根据其与拥有该技能的护士数 num 的比例确定先排哪个技能的所有时间槽. 如果 r/num 小, 说明拥有该技能的护士人手充足, 排班比较容易满足, 可以放到后面排; 反之说明拥有该技能的护士人手紧张, 应该优先考虑. 算法将从 r/num 比较大的技能开始, 给该技能的所有时间槽填充护士, 直到达到硬约束 H_2 规定的人数下限. 对于每个时间槽, 在保证不违反硬约束 H_1, H_3 和 H_4 的情况下, 优先选择拥有技能数更少的护士, 有多个技能数相同的护士时随机挑选一个, 以给后续技能的时间槽保留更充足的人选, 同时保证初始解的多样性. 例如第 d_1 天时间槽 (h_1,k_1) 至少需要一个掌握了技能 k_1 的护士, 护士 n_1 和 n_2 均只具备技能 k_1, 护士 n_3 同时具备 k_1 和 k_2 两项技能, 那么在挑选护士时, 从 n_1 和 n_2 中等概率随机选择一个填充该时间槽, 即安排 n_1 或 n_2 在第 d_1 天使用技能 k_1 上班次 h_1.

由于该贪心构造方法比较简单, 理论上可能生成不满足硬约束 H_2 的非法解. 但是由于软约束 S_1 的存在, 未达到人数下限意味着与最优人数差距较大, 会产生巨大的惩罚, 因此在禁忌搜索初期可以被轻松修复. 然而根据实验观察, 通过贪心构造方法得到合法初始解的频率为 100%, 不需要通过搜索修复.

3.3 邻域结构

对于一个给定的阶段 i 解向量 X_i, 可以通过对其施加一个动作得到一个新的解向量 X_i', 这个新的解向量称为 X_i 的邻域解, 使 X_i 变为某个邻域解 X_i' 的变换称为邻域动作 m, 令作用于 X_i 的邻域动作的集合为 $M(X_i)$; C 为 Boole 表达式到整数的转换函数, 如果所给 Boole 表达式为真取 1, 否则取 0. 则对于护士排班问题可以定义如下邻域结构.

- **增加排班.** 安排原本在第 d 天休息的护士 n 在第 d 天班次 h 技能 k 上班. 该邻域动作的集合为

$$M_1(X_i)=\{m_1(n,d,h,k)|\forall n\in N,\forall d\in D_i,\forall h\in H,\forall k\in K,x_{i,n,d}^H=\text{ShiftOff}\wedge h\neq\text{ShiftOff}\},\quad(3)$$

以图 1 所示的解向量为例, 将原本在 d_1 天休息的护士 n_4 安排到时间槽 (h_2,k_2) 工作, 则变为图 2 所示的邻域解.

- **减少排班.** 安排原本在第 d 天上班的护士 n 在第 d 天休息. 该邻域动作的集合为

$$M_2(X_i)=\{m_2(n,d)|\forall n\in N,\forall d\in D_i,x_{i,n,d}^H\neq\text{ShiftOff}\},\quad(4)$$

以图 1 所示的解向量为例, 安排原本在 d_1 天上班的护士 n_1 休息, 则变为图 3 所示的邻域解.

- **变更排班.** 安排原本在第 d 天班次 $x_{i,n,d}^H$ 技能 $x_{i,n,d}^K$ 上班的护士 n 在第 d 天班次 h 技能 k 上班. 该邻域动作的集合为

$$\begin{aligned}M_3(X_i)=&\{m_3(n,d,h,k)\mid\forall n\in N,\forall d\in D_i,\forall h\in H,\forall k\in K,x_{i,n,d}^H\neq\text{ShiftOff}\wedge h\neq\text{ShiftOff}\\&\wedge(x_{i,n,d}^H\neq h\vee x_{i,n,d}^K\neq k)\},\end{aligned}\quad(5)$$

以图 1 所示的解向量为例, 将第 d_6 天原本在时间槽 (h_2,k_2) 上班的护士 n_1 变为在 (h_1,k_1) 上班, 则变为图 4 所示的邻域解.

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	h_3, k_2		h_1, k_1	h_1, k_2	h_1, k_1	h_2, k_2	h_2, k_2
n_2	h_2, k_1	h_2, k_2			h_2, k_2	h_2, k_1	h_2, k_1
n_3	h_3, k_1	h_3, k_2	h_3, k_1	h_3, k_1			
n_4	h_2, k_2			h_3, k_2	h_3, k_2	h_3, k_2	h_3, k_2
n_5	h_1, k_2	h_1, k_2	h_2, k_2	h_2, k_2		h_1, k_2	h_1, k_2

图 2　增加排班示例

Figure 2　Example of adding assignment

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1			h_1, k_1	h_1, k_2	h_1, k_1	h_2, k_2	h_2, k_2
n_2	h_2, k_1	h_2, k_2			h_2, k_2	h_2, k_1	h_2, k_1
n_3	h_3, k_1	h_3, k_2	h_3, k_1	h_3, k_1			
n_4				h_3, k_2	h_3, k_2	h_3, k_2	h_3, k_2
n_5	h_1, k_2	h_1, k_2	h_2, k_2	h_2, k_2		h_1, k_2	h_1, k_2

图 3　减少排班示例

Figure 3　Example of removing assignment

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	h_3, k_2		h_1, k_1	h_1, k_2	h_1, k_1	h_1, k_1	h_2, k_2
n_2	h_2, k_1	h_2, k_2			h_2, k_2	h_2, k_1	h_2, k_1
n_3	h_3, k_1	h_3, k_2	h_3, k_1	h_3, k_1			
n_4				h_3, k_2	h_3, k_2	h_3, k_2	h_3, k_2
n_5	h_1, k_2	h_1, k_2	h_2, k_2	h_2, k_2		h_1, k_2	

图 4　变更排班示例

Figure 4　Example of changing assignment

- **交换排班.** 即块交换. 交换两个护士 n_1 和 n_2 在第 d_1 到 d_2 天的排班. 该邻域动作的集合为

$$M_4(X_i)=\left\{m_4(n_1,n_2,d_1,d_2)\,\middle|\,\forall n_1,n_2\in N,\forall d_1,d_2\in D_i,d_1\leqslant d_2\wedge\left(\sum_{d=d_1}^{d_2}\mathrm{C}(x^H_{i,n_1,d}\neq x^H_{i,n_2,d})\right)>0\right\},\quad(6)$$

以图 1 所示的解向量为例, 交换护士 n_1 和 n_2 从 d_2 到 d_5 天的排班, 则变为图 5 所示的邻域解.

4 种邻域结构中, 增加、减少和变更排班的复杂度相对较小, 而交换排班的复杂度较大. 虽然多个前 3 种邻域动作的组合可以构成块交换的邻域动作, 但局部搜索每一步通常选择能够改进目标函数的邻域动作, 可能错过可以产生更大改进的组合动作. 例如两个护士在某一天都上班时, 交换他们在这一天的排班可以等效成连续执行两个变更排班邻域动作. 考虑这个组合动作整体对目标函数值的改变, 可能是有较大改进的, 但是先单考虑其中一个护士的变更班次时, 会发现原班次的最优人数等约束的违反导致解的质量变差了, 于是该变更班次不会被执行, 从而导致组合动作不会被考虑. 另一方面, 如

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	h_3, k_2	h_2, k_2			h_2, k_2	h_2, k_2	h_2, k_2
n_2	h_2, k_1		h_1, k_1	h_1, k_2	h_1, k_1	h_2, k_1	h_2, k_1
n_3	h_3, k_1	h_3, k_2	h_3, k_1	h_3, k_1			
n_4				h_3, k_2	h_3, k_2	h_3, k_2	h_3, k_2
n_5	h_1, k_2	h_1, k_2	h_2, k_2	h_2, k_2		h_1, k_2	h_1, k_2

图 5 交换排班示例

Figure 5 Example of swapping assignments

果将复杂的邻域动作分解为若干个简单的邻域动作, 其中间状态可能违反硬约束. 例如将交换两个护士在某一天的排班分解为先后变更两个护士的排班, 可能出现先变更一个护士的排班后原时间槽的人数小于人数下限的情况. 因此, 交换排班作为前 3 种邻域动作组合而成的复合邻域结构, 具有极其重要的意义.

3.4 多邻域禁忌搜索

多邻域禁忌搜索以禁忌搜索为框架, 在搜索的每一步迭代中, 通过概率选择 4 种结构不同的邻域中的一种进行搜索. 如算法 3 所示, 该过程首先令选中各种邻域的概率相等, 然后将禁忌表初始化为允许所有邻域动作执行的状态. 接下来循环执行邻域搜索过程, 直到连续一定迭代步数 th 无法改进本次局部搜索找到的最优解 s^* 即停止. 在循环体中, 首先按概率在 $M_1 \sim M_4$ 中选择一个邻域 $\mathcal{M}$, 然后对 $\mathcal{M}$ 中的所有动作进行评估, 计算其对目标函数值的改变量, 即执行邻域动作后得到的解的目标函数与执行之前的解的目标函数的差值. 选出差值最小的最优邻域动作 m 之后, 更新相关的禁忌表项, 执行该邻域动作, 并更新本次搜索的最优解. 最后根据该邻域动作的优度更新下一次迭代中各邻域被选中的概率. 接下来将介绍上述过程的具体实现.

算法 3 多邻域禁忌搜索算法伪代码

Input: 初始解 s^0, 各护士的惩罚权重 w;
Output: 求得的当前权重下本阶段的最优排班计划 s^*;

```
s ⇐ s^0, s^* ⇐ s^0 // s 表示当前解
InitializeNeighborhoodSelectionPossibility(p_1, p_2, p_3, p_4); // p_1 ∼ p_4 分别表示选中 M_1 ∼ M_4 的概率
repeat
    M ⇐ SelectNeighborhood(p_1, p_2, p_3, p_4); // 按概率分布选择一种邻域, 见 3.4.1 小节
    m ⇐ FindBestNeighborhoodMove(s, M); // 禁忌搜索最优动作, 见 3.4.2 小节
    s ⇐ ApplyNeighborhoodMove(s, m); // 对当前解 s 执行动作 m
    UpdateOptima(s, s^*); // 更新最优解 s^*
    UpdateNeighborhoodSelectionPossibility(p_1, p_2, p_3, p_4, M, m); // 更新邻域选择概率, 见 3.4.1 小节
until 满足停止条件
```

3.4.1 邻域选择策略

由于本算法设计了 4 种结构不同的邻域, 局部搜索每一步对所有邻域进行探索开销过大, 因此每次只选择一个邻域进行评估. 于是, 邻域的选择策略的设计对算法性能的提升起到了至关重要的作用.

本算法采用了轮盘赌的策略, 每种邻域 M_i 有一定的概率 p_i 被选中, 所有邻域被选中的概率之和满足 $\sum_{i=1}^{4} p_i = 1$. 在局部搜索的每一次迭代中, 会在所有邻域中挑选一个邻域进行搜索, 寻找具有最大改进的邻域动作.

概率 p_i 由相应权重 ω_i 来控制, $p_i = \omega_i / \sum_{i=1}^{4} \omega_i$. 初始时 4 种邻域的权重相等, 即选中概率都为 0.25. 在搜索过程中, 算法将根据邻域 M_i 的表现动态调整其权重 ω_i. 每一迭代步搜索某一邻域寻找最优动作 m 时, 会出现 m 改进了最优解 s^*、改进了当前解 s、没有改进当前解以及不存在合法动作这四种情况之一. 这 4 种情况反映了该邻域对当前搜索过程的适应性, 表明了对解向量产生改进的程度. 很显然, 应该让更合适的邻域在接下来的迭代中以更大的概率被选择, 因此本算法设置了 4 个基准值 $\Omega_1 \sim \Omega_4$, 分别与上述 4 种情况相对应, 每执行一个邻域动作都将根据其效果调整对应邻域的权重向对应的基准值趋近, 其他邻域的权重保持不变. 具体的调整策略为

$$\omega_i = \begin{cases} \omega_i + \lambda(\Omega_1 - \omega_i), & m \text{ 改进了本次搜索的最优解;} \\ \omega_i + \lambda(\Omega_2 - \omega_i), & m \text{ 改进了当前解;} \\ \omega_i + \lambda(\Omega_3 - \omega_i), & m \text{ 没有改进当前解但是合法动作;} \\ \omega_i + \lambda(\Omega_4 - \omega_i), & \text{不存在合法动作.} \end{cases} \tag{7}$$

其中 $\Omega_1 > \Omega_2 > \Omega_3 > \Omega_4 > 0$, $\lambda \in (0, 1]$ 控制 ω_i 向基准值趋近的速度. 在其定义域内, λ 越大收敛速度越快. 例如, 局部搜索的某一迭代步选择了增加排班邻域 M_1, M_1 的权重 ω_1 此时位于区间 (Ω_2, Ω_1), 若该邻域中的最优动作 m 能够改进最优解 s^*, 则根据式 (7), M_1 的权重调整为 $\omega_1 = \omega_1 + \lambda(\Omega_1 - \omega_i)$, M_1 的权重增加, 因此之后的选中概率也将变大; 而如果该邻域不存在合法动作, 则权重调整为 $\omega_1 = \omega_1 + \lambda(\Omega_4 - \omega_i)$, M_1 的权重减少, 下一迭代步被选中的概率也将变小.

3.4.2 禁忌策略

按概率选定一种邻域后, 本算法采用了禁忌搜索策略以避免搜索过程陷入局部最优陷阱.

禁忌搜索的核心思想在于记录最近一定步数内已搜索过的解的特征或已执行过的邻域动作的特征, 防止搜索路径在局部最优附近绕圈. 一个护士在某一天的状态只存在两种可能: 休息或在某个时间槽上班, 当搜索中某一迭代步一个护士某一天的状态发生了改变, 那么在接下来的若干步内将避免该护士在这一天被安排回原先的状态. 具体来说, 本算法对排班情况进行了记录, 如果某一迭代步一个在第 d 天休息的护士 n 被安排上班, 那么在接下来的 l_1 步内, 将不能对护士 n 执行第 d 天的减少排班动作, 即不能在第 d 天被重新安排为休息; 如果某一迭代步一个在第 d 天班次 h 技能 k 上班的护士 n 被安排为休息或者在其他时间槽上班, 那么在接下来的 l_2 步内, 护士 n 将不能被重新安排在第 d 天的时间槽 (h, k) 上班. 在每一迭代步中, 只考虑邻域当中没有被禁忌的动作, 但如果一个动作可以改进本次局部搜索的最优解 s^* 则不考虑其是否被禁忌 (特赦准则). 最终在所有未禁忌的动作和满足特赦准则的禁忌动作中挑选出一个对目标函数值改进幅度最大的动作.

3.4.3 缓存策略

由于护士排班问题软约束众多, 目标函数计算过程十分复杂. 对于较大的邻域, 邻域评估的运算量更加庞大. 分析局部搜索每次只对当前解做小幅改动的特性, 可知每次执行一个邻域动作后, 大多数其他邻域动作对目标函数的影响并未发生变化. 例如, 将一个在第 d 天休息的护士 n_1 安排到时间槽 (h_1, k_1) 工作的增加排班动作 $m_1(n_1, d, h_1, k_1)$ 与安排一个在第 d 天时间槽 (h_2, k_2) 上班的护士 n_2

休息的减少排班动作 $m_2(n_2,d)$ 是互不影响的, 即不管其中一个动作执行与否, 另一个动作的执行对目标函数带来的变化不会发生改变. 因此, 本算法使用了缓存机制, 将邻域动作导致的目标函数的增量保存起来, 加速邻域评估过程.

本算法中有多种邻域结构, 而搜索过程中每次只挑选一个进行搜索, 如果对所有邻域都进行缓存, 会产生更大的缓存维护开销. 下面将分析对各种邻域进行缓存的利弊, 最终得出本算法使用的缓存策略.

当对护士 n 执行了 $M_1 \sim M_3$ 中的一个动作后, 由于其排班出现了变化, 所有与其相关的 4 种邻域 $M_1 \sim M_4$ 中的动作引起的目标函数的增量可能会发生改变. 同理, 当对护士 n_1 和 n_2 执行了块交换 M_4 后, 所有与护士 n_1 和 n_2 相关的 M_1 至 M_4 中的动作引起的目标函数的增量可能发生改变.

根据预先实验观察, 由于邻域 M_4 的规模远大于其他 3 种邻域, 对 M_4 进行邻域搜索的时间开销占用了总时间的 80% 以上, 对其进行缓存将极大地减少重复计算, 因此缓存该邻域动作引起的目标函数增量是非常有必要的. 而由上面的分析可以看出, $M_1 \sim M_3$ 的搜索时间开销较小, 并且所有邻域动作的执行都会影响其他邻域相关动作的缓存, 例如对护士 n 执行了增加排班动作后, 涉及护士 n 的减少排班动作、变更排班动作以及交换排班动作都会受到影响. 随着被缓存的邻域种类数的增多, 相互影响将被放大, 会有更多缓存在被再次利用之前就已经失效, 增加了缓存的维护开销.

因此, 经过上述综合分析, 本算法仅对 M_4 进行缓存, 对任意两个护士, 缓存其任意起止天的块交换中的最优动作及其引起的目标函数增量, 即计算两个护士之间所有可能的交换动作, 记录其中最优的交换. 当某个护士 n 的排班发生改变时, 其与所有其他护士的最优块交换缓存失效, 需要重新计算以更新这些缓存.

3.5 惩罚权重调整策略

为了增加算法的疏散性, 本算法为护士设置了惩罚权重 w. 除 S_1 以外的所有软约束都是针对单个护士的排班考虑的, 各护士对这些软约束的违反程度的计算是相互独立的, 即可以将其进一步细分, 将软约束 S_j 在所有护士上的惩罚之和表示为 $f_j^{\mathrm{S}}(X_i) = \sum_{n\in N} w_n \times f_j^{\mathrm{S}}(X_{i,n})$, 其中 $f_j^{\mathrm{S}}(X_{i,n})$ 表示解 X_i 中护士 n 软约束 S_j 的惩罚, w_n 为算法引入的护士 n 的惩罚权重. 在原始问题中, 可看作所有护士的权重 w_n 相等 (即 $w_n = 1, n \in N$), 而本算法将动态调整该权重. 集中性搜索前, 将所有护士的权重设置为 1, 当集中性搜索无法继续改进最优解时, 调整护士的权重进行疏散性搜索. 本算法通过增加惩罚较多的护士的权重, 引导搜索优先改进这些护士的排班, 即通过改变目标函数来探索其他有改进潜力的解空间.

护士惩罚权重的具体调整过程为, 在全部护士中, 选择惩罚最多的 $r_1 \times |N|$ 个护士, 将其权重设为 W_1, 然后在剩余护士中, 随机选择 $r_2 \times |N|$ 个护士, 将其权重设为 W_2, 再将剩下的护士的权重设为 W_3. 其中有 $W_1 \geqslant W_2 \geqslant W_3 \geqslant 0, r_1, r_2 \in [0,1], r_1 + r_2 < 1$. 为了增强扰动效果, 算法在具体实现时参数的取值为 $W_1 = W_2 = 1, W_3 = 0$, 即只考虑排班格局较差的护士以及为增加多样性随机挑选的护士. 例如, 若 $r_1 = 0.2, r_2 = 0.05$, 则在调整权重时, 首先将惩罚最多的 $0.2 \times |N|$ 个护士的权重设为 1, 然后在剩余的护士中随机挑选 $0.05 \times |N|$ 个护士将其权重设为 1, 剩下没有选中的护士权重设为 0, 即不考虑这些护士的惩罚. 该调整过程的伪代码如算法 4 所示.

3.6 全局约束的近似评估

软约束 S_7 和 S_8 属于评估整个排班周期的约束, 只有当求解最后一个阶段的排班时才能被准确计

算法 4 惩罚权重调整过程

Input: 护士集合 N, 某个护士当前惩罚的计算函数 f^{S};
Output: 护士的惩罚权重 w;

```
L ⇐ SortNurseByPenalty(N, f^S); // L 是将全体护士按惩罚从大到小排列的序列
i ⇐ 1;
while i < |N| × r_1 do
    n ⇐ L_i;
    w_n ⇐ W_1;
    i ⇐ i + 1;
end while
while i < |N| do
    n ⇐ L_i;
    w_n ⇐ W_3;
    i ⇐ i + 1;
end while
i ⇐ 1;
while i < |N| × r_2 do
    n ⇐ SelectNurseRandomly(N);
    if w_n = W_3 then
        w_n ⇐ W_2;
        i ⇐ i + 1;
    end if
end while
```

算, 但是如果先前阶段排班不考虑这些约束的话, 会严重影响最终全周期的排班效果, 甚至导致后续某阶段无法找到满足硬约束的排班计划. 因此, 在各阶段内部需要一个近似评估机制来估算本阶段的排班会对全局约束 S_7 和 S_8 造成多大的影响. 本算法在两个部分使用了近似评估策略, 其一是在计算本阶段的目标函数时, 根据各护士的总排班数上下限、历史排班数和本阶段人员需求等因素估算本阶段排班对 S_7 和 S_8 的贡献; 其二是在更新最优解时, 根据本阶段的排班情况设计辅助目标函数, 处理目标函数值相同的情况下最优解的取舍. 本节将对这两种策略进行详细介绍.

首先, 对于 S_7 的上限, 本算法采用了均匀分配剩余排班数上限的处理方案. 令 $A_{n,i}$ 为护士 n 在第 i 阶段的总排班数, 全周期总排班数下限为 A_n^{L}, 上限为 A_n^{U}, 则可以按如下公式模拟护士 n 在第 i 阶段的排班数上限

$$A_{n,i}^{\mathrm{U}} = \frac{A_n^{\mathrm{U}} - \sum_{j=1}^{i-1} A_{n,j}}{t-i+1}, \quad i = 1, 2, \ldots, t, \tag{8}$$

其中分子为剩余排班数上限, 由总排班数上限减去第 i 阶段之前的累计排班数求得, 其取值可能为负数, 即已经超过全周期的上限; 分母为包括阶段 i 在内的剩余阶段数. 例如, 求解一个全周期长度为 4 周的排班, 单阶段长度为一周, 护士 n 的总排班数上限 A_n^{U} 为 22, 则求解第 1 周的排班时, 该护士这一阶段的排班数上限 $A_{n,1}^{\mathrm{U}} = 22/4 = 5.5$. 若该护士第 1 周有 6 天上班, 则超过了该阶段的排班数上限, 受到 0.5 个惩罚. 依照式 (8) 可得该护士第 2 周的排班数上限 $A_{n,2}^{\mathrm{U}} = (22-6)/(4-1) = 16/3$, 依次类推可得剩余阶段的排班数上限.

相比于均匀分配各阶段的排班数, 即 $A_{n,i}^{\mathrm{U}} = A_n^{\mathrm{U}}/t$, 该方法更具灵活性, 即如果前期需求不多, 则中后期会放松该约束, 剩余排班数上限较多的护士可以更多地排班; 如果前期需求过多, 则中后期会收紧该约束, 保证最终排班数不会超出上限太多. 对于 S_8 所规定的上限, 采用了与 S_7 中的上限相同的近似评估方式.

其次, 本算法对 S_7 中总排班数下限的处理为在整个排班周期的前期若干阶段忽略该约束. 由于合同规定的总排班数下限是针对整个排班周期的, 而在开始的几个阶段中累积需求并不多, 所以该约束很容易被违反. 另外, 因为连续性约束 $S_2 \sim S_4$ 的存在, 各护士的排班可能出现不均衡的状态, 即在最初的几个阶段里, 某些护士排班比较满, 另外一些护士休息比较多. 介于上述两种情况, 在排班周期的前期阶段就考虑 S_7 的下限会鼓励算法增加总排班量, 这样易导致后期护士的总排班数大幅超出合同规定的上限. 因此, 本算法将根据 $A_n^{\mathrm{L}}/A_n^{\mathrm{U}}$ 确定从第几个阶段开始以均匀分配剩余排班数下限的方式考虑. 具体来说, 对于总共有 t 个阶段的排班周期, 将按如下规则计算护士 n 在第 i 个阶段的排班数下限

$$A_{n,i}^{\mathrm{L}} = \begin{cases} 0, & 1 \leqslant i < t \times \frac{A_n^{\mathrm{L}}}{A_n^{\mathrm{U}}}; \\ \frac{A_n^{\mathrm{L}} - \sum_{j=1}^{i-1} A_{n,j}}{t-i+1}, & t \times \frac{A_n^{\mathrm{L}}}{A_n^{\mathrm{U}}} \leqslant i \leqslant t. \end{cases} \tag{9}$$

例如一个护士 n 的总排班数上限 A_n^{U} 为 22, 下限 A_n^{L} 为 15, 则对于求解 4 周的排班来说, 前两周的排班数下限为 0, 即不需要考虑排班数下限, 从第 3 周开始考虑, 其计算方式与排班数上限相同.

最后, 考虑到对护士的需求往往超过了合同规定的工作量上限, 本算法中辅助目标函数的设计以优先选择排班数较少的方案为目标, 保证在不影响本阶段求解质量的情况下, 尽可能地为后续阶段预留人力资源. 对于两个目标函数值相同的解向量, 可能因为其对违反约束的组成的不同而对后续阶段的排班产生不同的影响. 例如, 有的排班可能对总排班数上限违反较多, 而对主要在本阶段内部发生作用的连续性约束和覆盖性约束违反很少, 这种排班会对后续阶段产生较大的负面影响, 反之, 可以给后续阶段留出更多的调整空间. 因此, 本算法中阶段 i 的解向量 X_i 的辅助目标函数将更倾向于保留总排班数较少的解. 其具体的计算方法为

$$f'(X_i) = \sum_{n=1}^{|N|} \frac{A_{n,i}}{2 \times A_n^{\mathrm{U}} - \sum_{j=1}^{i-1} A_{n,j}}, \quad i = 1, 2, \ldots, t. \tag{10}$$

由于不同护士的总排班数上限不同, 辅助目标函数并不是直接统计所有护士的总排班数, 而是统计各护士当前排班数占剩余排班数上限的比例之和, 保证排班的均衡性. 此外, 分母中对总排班数上限乘以 2 是为了保证分母为正数. 当两个解的目标函数值相等时, 优先选择辅助目标函数值更小的解.

4 计算结果

4.1 算例与运行环境

第 2 届国际护士排班竞赛的算例包括初赛的 14 个数据集和决赛的 6 个数据集. 所有数据集均由 1 个场景信息文件、3 个初始历史信息文件和 10 个阶段需求信息文件组成. 数据集基本数据由在多家 Belgium 医院工作过、对护士排班问题有丰富经验的人员提供生成.

其中, 场景信息文件包括以下内容:

- **周期长度.** 排班周期长度, 4 周或 8 周.
- **技能列表.** 列出所有技能名称.
- **合同列表.** 列出所有合同, 对于每一类合同, 给出该合同的名称, 规定的总排班数上下限, 连续工作天数上下限, 连续休息天数上下限, 总周末排班数上限以及是否要求完整周末.
- **护士列表.** 列出所有护士, 对于每一个护士, 给出该护士的名称, 所属合同以及具备的技能.

• **班次列表.** 列出所有班次, 对于每一种班次, 给出该班次的名称, 连续上该班次的天数的上下限以及该班次非法后继班次.

阶段需求信息文件包括以下内容:

• **护士需求.** 给出该阶段每天每个时间槽的护士人数下限和最优人数.

• **护士请求.** 列出该阶段的护士请求, 每一项护士请求由一个三元组表示, 其中包括护士名称, 日期以及班次名称, 表示该护士在该阶段的某一天请求不上该班次.

历史信息文件包括以下内容:

• **边界数据.** 列出所有护士的边界数据, 对于每一个护士, 给出该护士上阶段最后一天的工作班次 (如果最后一天休息, 用特殊标志 NONE 表示), 上阶段最后连续工作天数, 连续相同班次工作天数以及连续休息天数.

• **累积数据.** 列出所有护士的累计数据, 对于每一个护士, 给出该护士当前的累积排班数以及工作周末数 (在初始历史信息文件中这两项都是 0).

对于每个数据集, 数据规模主要与护士数相关, 而数据集中的护士数主要分布于 30 ∼ 120 之间. 初赛数据集的护士数包括 30, 40, 50, 60, 80, 100, 120, 周期长度包括 4 周和 8 周, 因此一共有 14 个数据集; 决赛数据集的护士数包括 35, 70, 110, 周期长度同样包括 4 周和 8 周, 因此一共有 6 个数据集. 不同的数据集关于护士掌握的技能、合同规定、每阶段护士人数需求、每阶段护士请求等存在不同之处. 具体运行时从中选取 1 个场景信息文件、1 个初始历史信息文件和 4 个或 8 个阶段需求信息文件的排列构成一个算例, 因此一个数据集可以生成大量算例. 算例的名称中包括了护士数、周数、初始历史文件编号和阶段需求文件编号序列, 例如, n035w4_2_8-8-7-5 表示有 35 个护士, 排班周期为 4 周, 初始历史文件编号为 2, 4 周的需求分别由编号为 8, 8, 7, 5 的需求文件提供. 这些文件按算法 1 所示的流程依次输入求解程序. 求解护士排班问题得到的目标函数值越小说明解向量的质量越好. 本次竞赛使用的算例[3)]与排名[4)]已公布于竞赛官方网站.

本算法使用 C++ 编写, 由 Visual C++ 2013 编译. 程序的运行环境为 Windows Server 2012 操作系统, Intel Xeon E5-2609 2.5 GHz CPU, 32 GB 内存. 运行时间由竞赛官方网站提供的基准测试程序[5)]求得, 其给定的运行时间与护士数量成正比. 算法中的重要参数如表 1 所示.

4.2 第 2 届国际护士排班竞赛结果对比

决赛中一共使用了 3 种规模的数据集共 60 个算例, 包括 35 个护士 4 周、35 个护士 8 周、70 个护士 4 周、70 个护士 8 周、110 个护士 4 周、110 个护士 8 周各 10 个算例, 对每个算例独立运行 10 次. 本算法在决赛中取得了第 4 名的成绩. 相比于其他进入决赛的队伍, 前两名在大多数算例上保持了一定的优势, 但是都存在无法于给定时间内求得合法解的情况, 其中第 1 名的算法在组委会提供的环境下进行的 600 次运行中, 有 34 次产生了非法解, 而本算法均能产生合法解, 表明本算法在稳定性上与前两名相比有一定的优势.

决赛中小规模、中等规模和大规模算例的计算结果分别如表 2∼4 所示, 表 2 给出了 35 个护士 4 周和 8 周共 20 个算例的结果, 表 3 给出了 70 个护士 4 周和 8 周共 20 个算例的结果, 表 4 给出了 110 个护士 4 周和 8 周的结果. 表格的内容为目标函数值, 其中 Known best 表示所有参赛者中的最优结果, Our best 表示本算法 10 次测试中的最优结果, Winner's best 表示决赛第 1 名获得者 10 次测试

3) 数据集和算例可在 http://mobiz.vives.be/inrc2/?page_id=20 下载.
4) 决赛排名可在 http://mobiz.vives.be/inrc2/?page_id=241 查看.
5) 基准测试程序可在 http://mobiz.vives.be/inrc2/?page_id=245 下载.

表 1 重要参数设置

Table 1 Settings of important parameters

Parameter	Section	Description	Value in this paper	Recommending range
th	3.4	Maximum iterations without improvement	scale[a)]	[0.8scale, 2scale]
Ω_1	3.4.1	Benchmark value in neighborhood selection	4096	$[1.5\Omega_2, 4\Omega_2]$
Ω_2	3.4.1	Benchmark value in neighborhood selection	2048	$[1.5\Omega_3, 4\Omega_3]$
Ω_3	3.4.1	Benchmark value in neighborhood selection	1024	$[1.5\Omega_4, 4\Omega_4]$
Ω_4	3.4.1	Benchmark value in neighborhood selection	256	$[0, 4096]$
λ	3.4.1	Adjusting factor in neighborhood selection	0.125	$[0.0625, 0.25]$
l_1	3.4.2	Tabu tenure	$1.5\|D_i\|$	$[\|D_i\|, 2\|D_i\|]$
l_2	3.4.2	Tabu tenure	$1.5\|D_i\|$	$[\|D_i\|, 2\|D_i\|]$
r_1	3.5	Nurse selection proportion	0.2	$[0.1, 0.4]$
r_2	3.5	Nurse selection proportion	0.05	$[0.01, 0.1]$

a) Scale stands for problem scale, and scale $= |N| \times |D_i| \times \sqrt{|H| \times |K|}$.

表 2 小规模算例计算结果

Table 2 Computational result on small scale instances

Instance	Known best	Our best	Winner's best	Our average	Winner's average
n035w4_2_8-8-7-5	1255	1335	1255	1370.5	1367.5
n035w4_0_1-7-1-8	1630	1685	1630	1756.5	1630
n035w4_0_4-2-1-6	1800	1970	1810	2021.5	1831.5
n035w4_0_5-9-5-6	1755	1760	1755	1834.5	1755
n035w4_0_9-8-7-7	1540	1660	1545	1723.5	1586
n035w4_1_0-6-9-2	1500	1650	1525	1737	1545
n035w4_2_8-6-7-1	1490	1600	1510	1644.5	1510
n035w4_2_9-2-2-6	1705	1895	1705	1947.5	1708
n035w4_2_9-7-2-2	1650	1940	1650	1970.5	1695
n035w4_2_9-9-2-1	1620	1880	1620	1927.5	1652
n035w8_0_6-2-9-8-7-7-9-8	3020	3505	99999	3628	99999
n035w8_1_0-8-1-6-1-7-2-0	2770	3405	2900	3653.5	12669.4
n035w8_1_0-8-4-0-9-1-3-2	2775	3190	2870	3378.5	80576.7
n035w8_1_1-4-4-9-3-5-3-2	2805	3180	2810	3325	2849.5
n035w8_1_7-0-6-2-1-1-1-6	2840	3440	2840	3548.5	2842
n035w8_2_2-1-7-1-8-7-4-2	2910	3445	3050	3672	90304.1
n035w8_2_7-1-4-9-2-2-6-7	2960	3540	2960	3632.5	3028.5
n035w8_2_8-8-7-5-0-0-6-9	2815	3385	2815	3603	2863
n035w8_2_9-5-6-3-9-9-2-1	3045	3350	3045	3533.5	3083.5
n035w8_2_9-7-2-2-5-7-4-3	2715	3295	2865	3488	2928

中的最优结果, Our average 表示本算法 10 次测试的平均值, Winner's average 表示决赛第 1 名获得者 10 次测试的平均值. 对于求出非法解的情况, 竞赛组委会将其目标函数值设为一个大于所有参赛者所求目标函数值的数值, 在本次竞赛中为 99999. 从结果对比可以看出, 本算法的最优解质量离第 1 名有

表 3　中等规模算例计算结果

Table 3　Computational result on medium scale instances

Instance	Known best	Our best	Winner's best	Our average	Winner's average
n070w4_0_3-6-5-1	2700	3090	2705	3151	2723
n070w4_0_4-9-6-7	2430	2790	2430	2889	2446
n070w4_0_4-9-7-6	2475	2865	2475	2948	2557.5
n070w4_0_8-6-0-8	2435	2855	2435	3016	2477
n070w4_0_9-1-7-5	2320	2725	2320	2864	2323
n070w4_1_1-3-8-8	2700	3035	2700	3134.5	2728
n070w4_2_0-5-6-8	2520	2880	2520	3012	2533
n070w4_2_3-5-8-2	2615	3050	2615	3141.5	2635
n070w4_2_5-8-2-5	2540	2875	2540	3005.5	2544.5
n070w4_2_9-5-6-5	2615	2975	2615	3046	2652
n070w8_0_3-3-9-2-3-7-5-2	5115	6000	5115	6222	5164
n070w8_0_9-3-0-7-2-1-1-0	5390	6420	5390	6602	5478.5
n070w8_1_5-6-8-5-7-8-5-6	5475	6095	5475	6236.5	5549
n070w8_1_9-8-9-9-2-8-1-4	5100	5700	5100	6018.5	5167
n070w8_2_4-9-2-0-2-7-0-6	5410	5990	5410	6259	5581.5
n070w8_2_5-1-3-0-8-0-5-8	5280	5975	5280	6315	5359.5
n070w8_2_5-7-4-8-7-2-9-9	5505	6210	5505	6317.5	5531.5
n070w8_2_6-3-0-1-8-1-5-9	5120	5960	5120	6255	5240
n070w8_2_8-6-0-1-6-4-7-8	5350	6205	5350	6492.5	7138.5
n070w8_2_9-3-5-2-2-9-2-0	5320	5895	5320	6044.5	5374

一定差距, 存在改进空间, 但是对不同算例均能提供合法解. 而第 1 名的算法在算例 n035w8_0_6-2-9-8-7-7-9-8 上 10 次运行均没有求出合法解, 同时在多个算例上产生了非法解 (n035w8_1_0-8-1-6-1-7-2-0, n035w8_1_0-8-4-0-9-1-3-2, n035w8_2_2-1-7-1-8-7-4-2, n110w4_1_0-1-6-4, n110w4_2_0-2-7-0), 导致其在这些算例上的平均求解质量不如本算法.

5　分析与讨论

第 3 节中描述了算法的关键模块和策略, 本章将对本算法使用的策略与其他可能的策略进行对比分析, 以阐明选择这些方案的原因. 本节使用的算例均从决赛算例中挑选, 对于每个算例, 独立重复运行了 32 次.

5.1　邻域结构对比分析

虽然块交换动作 M_4 理论上可以由多个 $M_1 \sim M_3$ 组合而成, 但是其在本算法中仍起到了至关重要的作用. 为了分析排班交换动作对算法的影响, 分别在有无交换排班动作的条件下对算法进行了测试. 图 6 分别从迭代次数和运行时间两个角度展示了有无 M_4 时算法在单个阶段中的收敛情况, 所用数据来自于随机挑选的一个小规模算例 (n035w4_2_8-8-7-5)、一个中等规模算例 (n070w4_0_3-6-5-1) 和一个大规模算例 (n110w4_0_1-4-2-8) 的 32 次独立运行中对应迭代步数和时间刻度位置取样所得的平

表 4 大规模算例计算结果

Table 4 Computational result on large scale instances

Instance	Known best	Our best	Winner's best	Our average	Winner's average
n110w4_0_1-4-2-8	2710	3395	2720	3539	2725
n110w4_0_1-9-3-5	2920	3475	2970	3663	3065
n110w4_1_0-1-6-4	2850	3685	2920	3769	12690.4
n110w4_1_0-5-8-8	2820	3440	2895	3569.5	3210
n110w4_1_2-9-2-0	3345	3995	3380	4092	3425
n110w4_1_4-8-7-2	2805	3515	2805	3661	2855.5
n110w4_2_0-2-7-0	3005	3795	3175	3903.5	51625.5
n110w4_2_5-1-3-0	2925	3520	3025	3637.5	3095
n110w4_2_8-9-9-2	3415	3895	3470	4025	3502.5
n110w4_2_9-8-4-9	3135	3640	3335	3769	3540
n110w8_0_2-1-1-7-2-6-4-7	5155	6420	5165	6596	5243
n110w8_0_3-2-4-9-4-1-3-7	4805	6015	4830	6172.5	4982.5
n110w8_0_5-5-2-2-5-3-4-7	4750	6065	4870	6227	4939
n110w8_0_7-8-7-5-9-7-8-1	4855	6105	5005	6251.5	5234
n110w8_0_8-8-0-2-3-4-6-3	4465	5920	4955	6146.5	5021.5
n110w8_0_8-8-2-2-3-2-0-8	4865	6310	5435	6469	5510.5
n110w8_1_0-6-1-0-3-2-9-1	5090	6330	5175	6514	5259.5
n110w8_1_4-1-3-6-8-8-1-3	4315	5925	4785	6115.5	4842.5
n110w8_2_2-9-5-5-1-8-4-0	4770	5840	5200	6222.5	5274
n110w8_2_8-5-7-3-9-8-8-5	4360	5540	4765	5809	5292

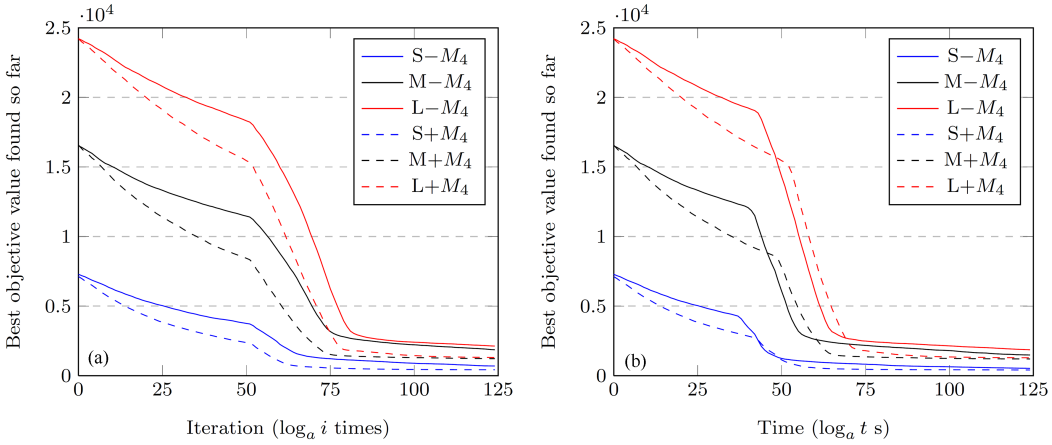

图 6 (网络版彩图) 块交换邻域的重要性

Figure 6 (Color online) Significance of block swap neighborhood. (a) Measursed by iteration; (b) measured by time

均值. 3 种规模的算例在图例中从小到大分别用 S, M 和 L 表示, 而是否使用 M_4 邻域用 + 和 − 表示. 例如, S−M_4 表示没有 M_4 的条件下算例 n035w4_2_8-8-7-5 的测试结果, S+M_4 表示有 M_4 的条件下算例 n035w4_2_8-8-7-5 的测试结果. 图 6(a) 的横坐标为搜索迭代步数, 纵坐标表示截止当前迭代步

算法所求得的单个阶段的最优目标函数值; 图 6(b) 的横坐标为算法执行时间, 纵坐标表示截止当前时间点算法所求得的单个阶段的最优目标函数值. 因为算法初期收敛速度非常快, 而后期目标函数值变化幅度较小, 导致等间隔取样的结果不便于观察比较. 故图中横坐标为对迭代步数和时间刻度取对数的结果, 分别为 $x = \log_a i$ (图 6(a)), $x = \log_a t$ (图 6(b)), 其中 $a = 1.08$, i 为取样的迭代步数点, t 为取样的时间点.

从图 6 中可以看出, 以迭代次数为取样间隔时, 块交换对算法优度的提升效果尤为明显, 即在相同的迭代步数下, 使用了交换排班动作的算法所得结果更优. 但是由于搜索交换排班动作邻域的时间开销较高, 即在时间相同的情况下, 增加了块交换之后搜索执行的步数更少, 因此在算法执行中期的某一时间段内, 优度不如没有使用块交换动作的算法. 但在搜索初期的快速改进阶段和后期的深度搜索阶段仍有较大优势, 且收敛的极限优于没有块交换的情况. 由此可见, 块交换邻域无论从短期的收敛速度来考虑, 还是从长期的目标函数值下限来考虑, 都发挥了重要的作用.

5.2 疏散性搜索策略对比分析

观察目标函数值的组成可以发现, 不同护士的惩罚有一定的差异. 然而由于众多约束相互制约, 局部搜索后期一旦落入局部最优就很难再进行结构性的调整, 而 3.5 小节所述权重调整策略 (Adjust weight) 可以打破既定的排班格局, 为算法提供一个有导向而非盲目的疏散性搜索过程. 为了验证其有效性, 设计了另外两种扰动策略进行对比. 其一是随机游走 (Random walk), 即执行一定步数的合法邻域动作, 每个动作从 4 种邻域中等概率选择的一个邻域中随机挑选而得, 不考虑动作的优度. 其二是部分重建 (Partial rebuild), 即随机保留解向量的部分元素, 然后对不满足人数下限的时间槽随机添加护士直至重新产生合法解. 对随机游走、部分重建和权重调整 3 种扰动方式进行了对比测试. 测试从决赛使用的算例中随机挑选了 18 个算例分别进行了 32 次独立的重复计算, 并统计了多次计算得到的目标函数值中的最优值和平均值. 表 5 给出了对比结果, 其中 Random walk 表示使用随机游走作为扰动策略, Partial rebuild 表示使用部分重建作为扰动策略, Adjust weight 表示使用权重调整作为扰动策略, Best 表示 32 测试中的最优结果, Average 表示 32 测试的平均结果.

从表 5 可以看出, 比较 32 次计算求得的最优值, 不同疏散性搜索策略没有显著差异, 不存在一种策略在大部分算例上优于另外两种策略的情况, 其中权重调整策略在 5 个算例上最优解质量优于另外两种策略. 而对比平均值却可以发现使用了权重调整策略之后有比较明显的提升, 在 75% 的算例上平均求解质量超过了另外两种扰动策略, 离算法所能求到的下限更加接近. 这说明本算法使用的权重调整策略是一种有效的疏散性搜索策略, 其有导向的扰动增加了算法的稳定性.

5.3 全局约束近似评估策略对比分析

为了分析全局约束 S_7 和 S_8 的近似评估策略对算法的影响, 设计对照实验比较了忽略全局约束 (Ignore)、采用均分各阶段排班数的策略 (Even allocation) 和 3.6 小节所述的均匀分配剩余排班数的策略 (EA on remainder) 的求解质量. 实验随机挑选了 6 个算例分别进行了 32 次独立的测试. 表 6 给出了分别使用这 3 种策略多次测试所得最优结果 (Best) 和平均结果 (Average) 的对比情况.

从表 6 可以看出, 完全忽略全局约束效果明显较差, 无论是从求得的最优解来看还是从多次运行的平均值来看都不如另外两种方案. 而采用均匀分配剩余排班数, 使先前阶段排班的紧凑程度能影响到后续阶段的排班之后, 在 5 个算例上最优结果和平均值都优于均匀分配各阶段排班数策略, 计算结

表 5 疏散性策略的计算结果对比

Table 5 Comparison of computational results between diversification strategies

Instance	Best			Average		
	Random walk	Partial rebuild	Adjust weight	Random walk	Partial rebuild	Adjust weight
n035w4_0_1-7-1-8	1675	1705	1685	1788.91	1778.28	1747.19
n035w4_1_0-6-9-2	1675	1650	1655	1791.56	1774.38	1747.03
n035w4_2_8-6-7-1	1520	1575	1555	1628.28	1656.25	1635.47
n035w8_0_6-2-9-8-7-7-9-8	3410	3315	3420	3665.00	3597.03	3578.28
n035w8_1_0-8-1-6-1-7-2-0	3400	3350	3370	3602.03	3616.88	3601.88
n035w8_2_2-1-7-1-8-7-4-2	3495	3420	3460	3769.84	3703.13	3682.50
n070w4_0_3-6-5-1	3115	3025	3010	3243.44	3170.78	3154.38
n070w4_1_1-3-8-8	3050	3040	3020	3118.75	3136.25	3128.44
n070w4_2_0-5-6-8	2895	2815	2890	3034.69	3021.41	3015.78
n070w8_0_3-3-9-2-3-7-5-2	6140	6035	5945	6415.94	6289.53	6230.16
n070w8_1_5-6-8-5-7-8-5-6	6105	5910	5940	6325.63	6219.22	6214.22
n070w8_2_4-9-2-0-2-7-0-6	6050	5865	5880	6373.44	6280.63	6210.94
n110w4_0_1-4-2-8	3340	3400	3355	3516.72	3565.94	3527.81
n110w4_1_0-1-6-4	3630	3615	3670	3765.31	3768.44	3794.38
n110w4_2_0-2-7-0	3745	3720	3760	3893.13	3907.66	3925.00
n110w8_0_2-1-1-7-2-6-4-7	6255	6300	6235	6595.16	6578.59	6532.97
n110w8_1_0-6-1-0-3-2-9-1	6310	6280	6270	6608.13	6512.66	6469.69
n110w8_2_2-9-5-5-1-8-4-0	5855	6115	6045	6257.34	6316.25	6300.94

表 6 全局近似评估策略的计算结果对比

Table 6 Comparison of computational results between approximate global constraints evaluation strategies

Instance	Best			Average		
	Ignore	Even allocation	EA on remainder	Ignore	Even allocation	EA on remainder
n035w4_2_8-6-7-1	2300	1610	1540	2505.31	1705.47	1642.81
n035w8_0_6-2-9-8-7-7-9-8	4700	3485	3410	5059.84	3695.47	3574.22
n070w8_2_8-6-0-1-6-4-7-8	8305	6390	6155	8725.31	6801.72	6502.03
n110w4_1_0-1-6-4	4930	3655	3630	5532.03	3837.50	3789.06
n110w4_2_0-2-7-0	5175	3685	3695	5693.59	3915.31	3931.88
n110w8_2_8-5-7-3-9-8-8-5	8810	5725	5565	9380.00	6047.34	5776.56

果相比于均匀分配各阶段排班数有大约 3% 的改进. 由此可见本算法使用的全局约束的近似评估策略的有效性.

6 总结

本文提出了一种用于求解多阶段护士排班问题的启发式算法. 在对第 2 届国际护士排班竞赛所提出的问题进行了简要的介绍之后, 本文详细描述了带权禁忌搜索算法. 算法以禁忌搜索为核心, 通

过调整护士的惩罚权重来实现搜索过程中集中性与疏散性的平衡. 算法设计了增加排班、减少排班、变更排班和交换排班 4 种邻域结构, 并使用了一种根据其适应性动态调整各邻域被选中的概率的策略. 为了解决全局约束无法在单个阶段计算的问题, 算法使用了一种近似评估方案来衡量各阶段的排班对全局约束产生的影响. 此外, 本文通过对比实验展示了不同邻域结构对收敛速度产生的影响, 得出了权重调整策略对提升求解质量具有重要意义的结论, 表明了全局约束的近似评估策略的合理性.

第 2 届护士排班竞赛的计算结果表明了算法的有效性. 在决赛的 60 个算例上, 算法在保证结果合法性的同时, 求解质量也接近目前已知的最优解, 最终在第 2 届国际护士排班竞赛中获得了全球第 4 名的成绩. 但是, 由于算法使用的邻域变化模式比较简单, 对于约束众多的问题, 容易陷入局部最优, 或者难以跨过非法解以探索更广阔的解空间. 因此可以考虑增加更大的邻域, 对解空间进行更充分的探索. 与此同时, 对全局约束的近似评估策略也可以有更细致的考量, 让算法对多阶段问题的求解更接近掌握了整个排班周期的完整信息的理想状态. 另外, 在允许多线程计算的环境下, 局部搜索中的每一迭代步可以通过并行评估邻域动作来加快搜索速度.

目前国内医院的护士排班基本依靠人工经验来完成, 排班一般为固定循环模式. 为国内医院护士排班问题提供智能化的解决方案还需要与国内医院做进一步的沟通和交流, 进行有针对性的需求分析和数学建模, 本文提出的算法可以作为后期在国内医院推广应用的理论基础.

值得注意的是, 带权禁忌搜索的思想并不局限于护士排班问题的求解, 其对解向量产生有潜力的结构性改变的能力, 可以在很多组合优化问题中得到应用, 为算法提供有导向的疏散性搜索. 而算法对多阶段问题的求解能力, 更接近大多数问题的现实情况. 对不确定因素和易变因素进行考虑, 将使理论问题的研究更具有实际意义和应用价值.

参考文献

1 Cheang B, Li H B, Lim A, et al. Nurse rostering problems—a bibliographic survey. Eur J Oper Res, 2003, 151: 447–460

2 Burke E K, de Causmaecker P, Berghe G V, et al. The state of the art of nurse rostering. J Scheduling, 2004, 7: 441–499

3 de Causmaecker P, Berghe G V. A categorisation of nurse rostering problems. J Scheduling, 2011, 14: 3–16

4 Isken M W. An implicit tour scheduling model with applications in healthcare. Ann Oper Res, 2004, 128: 91–109

5 Glass C A, Knight R A. The nurse rostering problem: a critical appraisal of the problem structure. Eur J Oper Res, 2010, 202: 379–389

6 Beliën J, Demeulemeester E. A branch-and-price approach for integrating nurse and surgery scheduling. Eur J Oper Res, 2008, 189: 652–668

7 He F, Qu R. A constraint programming based column generation approach to nurse rostering problems. Comput Oper Res, 2012, 39: 3331–3343

8 Burke E, de Causmaecker P, Berghe G V. A hybrid tabu search algorithm for the nurse rostering problem. In: Proceedings of Simulated Evolution and Learning. Berlin: Springer, 1999. 187–194

9 Burke E, Cowling P, de Causmaecker P, et al. A memetic approach to the nurse rostering problem. Appl Intell, 2001, 15: 199–214

10 Burke E, de Causmaecker P, Petrovic S, et al. Variable neighborhood search for nurse rostering problems. In: Proceedings of Metaheuristics: Computer Decision-Making. Berlin: Springer, 2004. 153–172

11 Burke E K, Curtois T, Post G, et al. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. Eur J Oper Res, 2008, 188: 330–341

12 Burke E K, Curtois T, Qu R, et al. A scatter search methodology for the nurse rostering problem. J Oper Res Soc, 2010, 61: 1667–1679

13 Burke E K, Curtois T, Qu R, et al. A time predefined variable depth search for nurse rostering. INFORMS J Comput,

2013, 25: 411–419
14 Bai R, Burke E K, Kendall G, et al. A hybrid evolutionary approach to the nurse rostering problem. IEEE Trans Evol Comput, 2010, 14: 580–590
15 Anwar K, Awadallah M, Khader A T, et al. Hyper-heuristic approach for solving nurse rostering problem. In: Proceedings of IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL), Orlando, 2014. 1–6
16 Huang H, Lin W J, Lin Z Y, et al. An evolutionary algorithm based on constraint set partitioning for nurse rostering problems. Neural Comput Appl, 2014, 25: 703–715
17 Haspeslagh S, de Causmaecker P, Schaerf A, et al. The first international nurse rostering competition 2010. Ann Oper Res, 2014, 218: 221–236
18 Ceschia S, Dang N T T, de Causmaecker P, et al. Second international nurse rostering competition (INRC-II)—problem description and rules—. arXiv:1501.04177, 2015
19 Valouxis C, Gogos C, Goulas G, et al. A systematic two phase approach for the nurse rostering problem. Eur J Oper Res, 2012, 219: 425–433
20 Burke E K, Curtois T. New approaches to nurse rostering benchmark instances. Eur J Oper Res, 2014, 237: 71–81
21 Lü Z P, Hao J K. Adaptive neighborhood search for nurse rostering. Eur J Oper Res, 2012, 218: 865–876

Weighted tabu search for multi-stage nurse rostering problem

Zhouxing SU, Zhuo WANG* & Zhipeng LÜ

The Laboratory of Smart Computing and Optimization, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074*, China*
*E-mail: wang_zhuo@hust.edu.cn

Abstract This paper presents a study of the multi-stage nurse rostering problem, which was proposed in the Second International Nurse Rostering Competition. We propose a weighted tabu search algorithm for solving this rostering problem, which is a very important problem in healthcare optimization. The algorithm employs the following neighborhood structures: 3 structures that are exclusive to each other and 1 compound structure. The possibilities of selecting a neighborhood to search are tuned dynamically by the extent to which they are adaptive. Meanwhile, the balance between intensification and diversification during the search procedure is achieved by adjusting the weight of the penalty for each nurse. We address the problem caused by the lack of global information within each independent stage by proposing an approximate evaluation strategy for spanning constraints. The algorithm employs a cache for neighborhood move evaluation with the purpose of reducing the overall computational cost for the adaptive neighborhood selection strategy, which improves the efficiency of the algorithm further. The benchmark result of our algorithm on 60 instances from the competition shows good solving quality and ranks fourth in the final of the Second International Nurse Rostering Competition. Furthermore, we compare and analyze some key factors in the algorithm and demonstrate their rationality.

Keywords nurse rostering, tabu search, timetabling, personnel scheduling, metaheuristic, combinatorial optimization

Zhouxing SU was born in 1992. He received his B.E. degree in computer science and technology from Huazhong University of Science and Technology, China, in 2014. He is currently working towards his M.E. in computer software and theory from the Laboratory of Smart Computation and Optimization (SMART), at the School of Computer Science and Technology of Huazhong University of Science and Technology. His research focuses on using metaheuristics to solve real-world applications such as personnel rostering, inventory routing, and the facility location problem.

Zhipeng LÜ was born in 1979. He received the B.S. degree in applied mathematics from Jilin University, China, in 2001, and the Ph.D. degree in computer software and theory from Huazhong University of Science and Technology, China, in 2007. He was a postdoctoral research fellow at LERIA, Department of Computer Science, University of Angers, France, from 2007 to 2011. He is a professor at the School of Computer Science and Technology of Huazhong University of Science and Technology and the director of the Laboratory of Smart Computing and Optimization (SMART). His research interests are in the areas of artificial intelligence, computational intelligence, green computing, and adaptive metaheuristics for solving large-scale real-world and theoretical combinatorial optimization and constrained satisfaction problems.

Zhuo WANG was born in 1990. He received the B.E. degree in computer science and technology from Huazhong University of Science and Technology, China, in 2012. He is currently working toward the Ph.D. degree in computer software and theory from the Laboratory of Smart Computing and Optimization (SMART), School of Computer Science and Technology of Huazhong University of Science and Technology. His research interests include load balancing in cloud computing, the generalized assignment problem, the graph coloring problem, and heuristic optimization.