Harmonious Personnel Scheduling

Gerhard Post and Bart Veltman

ORTEC bv, Groningenweg 6-33, 2803 PV Gouda, The Netherlands

1 Introduction

The area of personnel scheduling is very broad. Accordingly, planning problems range from assignment problems, along project scheduling problems, to nurse scheduling problems. We direct our attention to the "shift assignment" problem:

> **Given** shifts for a period, employees for (a part of) this period, qualifications of employees for shifts, preferences, requirements, ergonomic criteria
>
> **Assign** as many shifts as possible to the employees.

In the situations we encountered, the planning period is about a month, and the number of employees ranges from 20 to 40. Accordingly there are around 500 shifts to be planned.

Our aim is to discuss how HARMONY handles this planning problem. HARMONY is a commercial computer package, developed by ORTEC, and running in a variety of organizations, like ambulance services, refineries, security firms, and hospitals. We give an overview of the constraints that HARMONY can handle, the basic design of the algorithm, and discuss some results obtained by it.

2 Constraints

The constraints fall apart in two classes: hard constraints and soft constraints. We call the hard constraints *requirements*, and the soft constraints *criteria*. For each criterion, the user can adjust a weight. The requirements have their origins in legislation, quality of the schedule (ergonomic requirements), or personnel requests. The origins of the criteria are in ergonomic considerations, personnel requests, and day constraints. HARMONY knows around 60 requirements (of which some are very organization-specific), and around 20 criteria. Most constraints fall in the following classes:

- Restrictions on a single shift (skills, location of a shift, preference for a shift);
- Restrictions on availability for shifts (maximal work-time in a period, maximal number of night shifts, not available on certain days or certain times of day, number of shifts per week);
- Structure of series of shifts (minimal/maximal series length, combinations of consecutive shifts);

- Restrictions on rest time (daily rest time, weekly rest time, rest time after a night shift, or a night shift series);
- Restrictions for the weekend (no work in weekend/Sunday, no or otherwise 2 shifts in weekend, number of free weekends);
- Day constraints (quality of team at work, priority of shifts, cooperation)
- Constraints for on-call duties.

The user can provide criteria for the preferred situation per employee or group of employees. Usually, no two employees are the same: this holds for the social structures (which can imply certain constraints), the contracts, but also for personal preferences in shifts and series structures. Hence the number of requirements in use can be extensive; in real-life situations 100 requirements and constraints is normal.

3 Algorithmic approach

The algorithmic approach that is chosen is a highly random algorithm. The reason is apparent from the sections above: the initial planning problem is so broad, and the user can customize his planning problem in so many ways, that it will be very difficult to develop problem specific solvers. The algorithm consists of two steps:

1. **Genetic Algorithm Phase**
   The population consists of a (user controlled) number of individuals. Each individual is a schedule. The individuals in the initial population are constructed by dividing the employees at random in two groups, which are scheduled consecutively. By using cross-over and mutation operators, new individuals are constructed, as many as the size of the population. The next generation consists of the *best* individuals among the new individuals and members of the previous generation. Due to this greedy selection, the genetic algorithm phase converges (does not improve) after no more than 100 generations.
2. **Local improvement Phase**
   The best found schedule is taken, and locally optimized by two operators:
   - Exchange of two shifts. If the exchange is allowed (by the requirements) and an improvement (according to the criteria), keep the exchange.
   - Reassignment of a shift. If this is allowed, and an improvement, keep the reassignment.

   If the user puts no time limit, these operators are executed until a local optimum is reached.

At the moment that no improvement is found, we stop. Depending on the size of the instance, and the size of the population, this can take a few minutes, to several hours.

4 Discussion

Extensive testing was performed, especially on nurse scheduling type data. Several "local improvements" of the algorithm were tested, but none performed better. Nevertheless, the results are usually not optimal. It appears that problem specific difficulties (in nurse scheduling, for instance night shifts) are poorly solved. Two ways of handling these difficulties are proposed:

- Make that the user can indicate the major difficulties. In nurse scheduling, for instance, it is known that night shifts are often the bottleneck. So why not schedule these first? Even better, the algorithm should be able to detect bottlenecks, by prior analysis of the constraints, or by analyzing the results.
- One can notice that often the end result can be improved by (short) cycles of reassigning shifts. This suggests that a specific solver (tabu search?) can be of help.