Using Particle Swarm Optimization to Determine the Visit Times in Community Nurse Timetabling

Kevin Martin and Mike Wright

Department of Management Science, Lancaster University, LA1 4YX, United Kingdom

Abstract. In the problem of community nurse timetabling where the visits have accompanying time windows, knowing the order of the visits is not enough, the actual times of the visits need to be known. This is normally solved using a simple greedy approach, but the solutions reached this way can exhibit serious flaws . We show how the determination of these times can be modelled as a convex optimization problem, and demonstrate how this can be solved using Particle Swarm Optimization. Initial results show that this approach is far superior to the greedy approach.

1 Introduction

A very important aspect of mental health services is the provision of nurses to clients living in the community. This is a very complex procedure which needs to be done well so as to give the best possible care without exceeding financial budgets.

When timetabling these nurses, there are several constraints (some hard, some soft) and preferences to consider. These include (not in any particular order):

- Required durations of visits.
- Travel times between clients.
- Time windows (either constraints or preferences).
- Nurses must have the necessary skills to deal with the clients' needs.
- Clients may need two nurses working together.
- Clients' preferences - the provision of the "right" nurse can be a very strong preference, though usually not a hard constraint.
- Employee preferences, important to ensure good staff retention rates.
- Costs of travel and overtime.

This is not an exhaustive list.

The problem is a cross between employee timetabling and vehicle routing. Very few researchers have considered this combination. One of the first major attempts to solve the problem was done by Eveborn et al.[1] However their model conflicts with some of our objectives.

2 Model

Since no soft constraint or preference is overwhelmingly more important than any other, they are considered all together in an optimization problem with several weighted sub-objectives, giving an overall cost function to be minimised. Final determination of weights will depend upon consultation with the customers and the results of experimentation, to ensure that effective practical timetables are achieved.

The finished software not only needs to provide timetables from scratch but must also be usable to repair situations when sudden unexpected changes occur, such as staff sickness. Possibilities are needed within a matter of minutes and may involve the relaxation of some hard constraints - which the timetabler will have to choose. For these reasons a VRP based local search is used.

The model has an outer loop and an inner loop. The outer loop being the local search procedure, which defines solutions as an ordered list of visits for each nurse. Considerable work is ongoing to ensure that this procedure is highly effective and efficient. The order of visits is insufficient to specify the exact timetable, or critically, its cost. For this the precise times of each visit must be determined, this is done in the inner loop. This paper discusses the inner loop algorithm, and the advantages it provides to the outer loop.

3 Determining Visit Times

A common method for determining visit times for problems with time windows is a greedy approach. For example, events are considered in topological order of their dependencies and each event is timetabled to start at the earliest possible time, or at the start of the time window, whichever is later.

However, this will frequently give a severely suboptimal(by many objectives) solution. For example, the time window for a visit early in a nurse's schedule may be later than that for several visits later in the schedule. The greedy approach will place the former visit within its time window, with the result that the later visits will be timetabled considerably after their windows. It would be better for the former event to be timetabled well before its time window in order to allow for the later visits to be timed more appropriately. The model used in the inner loop therefore needs to be more complex than a simple greedy approach.

3.1 Flat Linear Approach

Consider an ordered set of visits $\{V_i\}$ with durations $\{D_i\}$, with time windows starting at $\{A_i\}$ and ending at $\{B_i + D_i\}$; thus ideally the start time S_i of event V_i will lie between A_i and B_i. Denoting the distances of the start from the sides of the window as α, and β, our aim is to minimise the function:

$$\sum w_i(\alpha + \beta) \text{ where } \alpha = |A_i - S_i| \text{ and } \beta = |B_i - S_i|$$

subject to the visits being carried out in the required order and with sufficient time to travel between visits. The set of weights $\{w_i\}$ can be altered to reflect the relative importance of visits being within their allotted time windows.

This objective function was chosen because when $A_i \leq S_i \leq B_i$ (Fig 1) the function achieves its minimum with a value of $B_i - A_i$. As S_i moves a distance δ out of this range (Fig 2) then the function increases linearly with δ. This objective can be minimised by Linear Programming, however, slight changes to it make LP unusable.

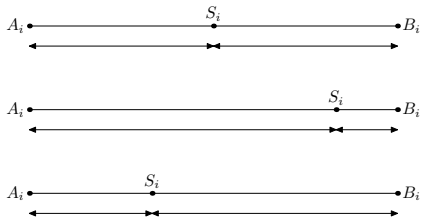

Fig. 1. If S_i is in window

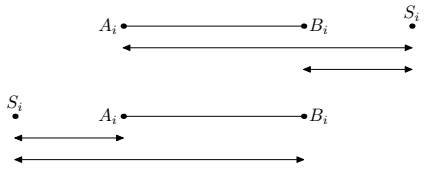

Fig. 2. If S_i is not in window

3.2 Other Objectives

The outer loop(local search) has control over the domain of the subproblem. The local search decides the order in which the visits should occur which directly corresponds to the constraints and hence the domain of the subproblem.

Bot the local search and the subproblem have information that could be useful to the other. The subproblem knows of gaps in the schedule where it would be good to insert new events and also of places where it would be very bad. The local search knows how long working days should be, to whom events should be assigned, etc... It therefore seems a good idea that there should be a high level of communication between the local search and the subproblem.

One way this can happen is by the solution of different objectives, objectives that the local search can dynamically construct. This is more than just setting

weights, it is the construction of specific objectives whose solution helps the local search make a decision.

Possible uses of dynamically constructed objectives include:

- Determining if the order of events should be changed.
- Finding gaps of specified size to insert new events.
- Penalising with nonlinear functions eg logarithmic or exponential.

Many of the objectives that could be constructed could be solved by an exact traditional technique, and due to the small instance size of the problem, could also be done quite quickly. However, such techniques require knowledge about the objective, and may even need to be tailored around the objective's structure. This isn't possible as the objectives are being constructed at run time. We therefore need a system which will work well with little/no knowledge of the function it is optimising.

4 Particle Swarm Optimization

The system is built around Particle Swarm Optimisation (PSO), a technique developed by Kennedy & Eberhart[2] for continuous optimisation problems. In PSO, a Discrete time simulation of a particle system is run. Particles move around solution space with velocities determined by the positions of the best solution found by that particle and the globally best solution found, incorporating a random element. A great advantage of this is that no knowledge of the objective function or constraints is required. Therefore it is very useful for this work. It is a heuristic method that does not guarantee a precise optimum, but this is not important to us as long as the solutions are quite close to the optimum.

There appears to have been no previous reported use of PSO for genuinely continuous problems arising in timetabling; however the continuous approximation of a very similar problem has been solved using PSO by Akjiratikarl and Yenradeea[3].

The results from the inner loop are cached and the PSO only runs on altered data. Here its time complexity is $O(Npd)$, where N is the number of iterations, p the number of particles and d the number of visits in the calculation. The time constant can be made very small with expert programming, especially on computers supporting vector mathematics and threading.

Initial experiments show that with the flat linear objective, PSO solutions are significantly better than greedy solutions Fig 3 and on average are within 1% of the optimal solution. Further experimentation will determine the most effective values of N and p, to achieve the best trade-off between time taken and solution quality. Effective methods for setting initial particle positions and momentums are also being investigated.

The presentation will describe the PSO implementation and experimental results.

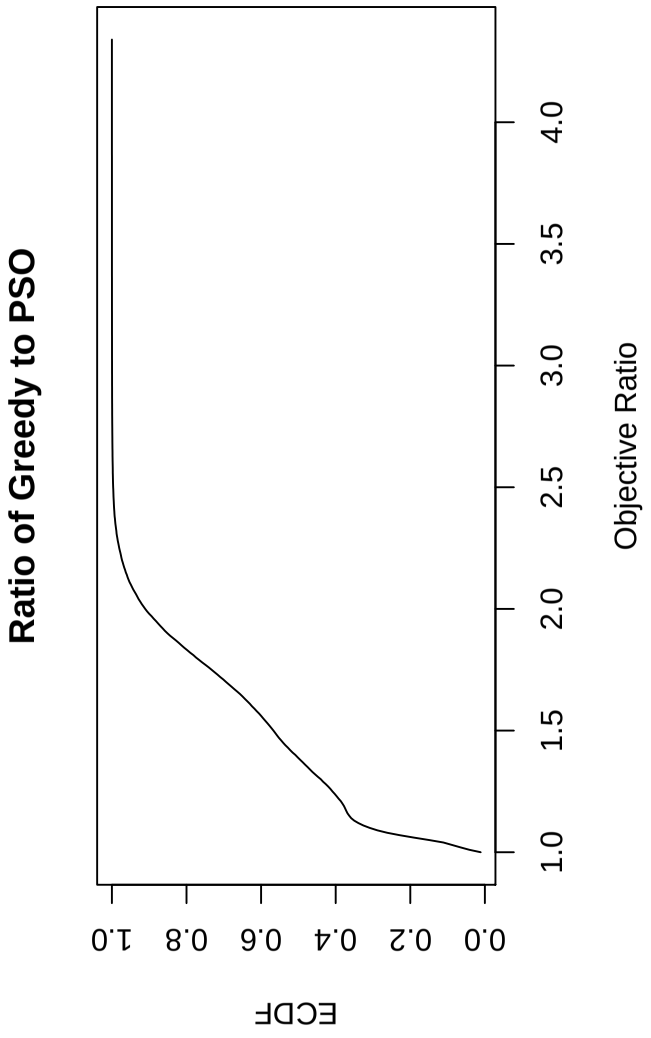

Fig. 3. Greedy Percentage Above PSO (Flat Linear Objective)

References

1. Eveborn, P., Flisberg, P., Ronnqvist, M.: Laps care–an operational system for staff planning of home care. European Journal of Operational Research **171**(3) (2006) 962–976
2. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Neural Networks, 1995 Proceedings, IEEE International Conference on. Volume 4. (1995) 1942–1948
3. Akjiratikarl, C., Yenradee, P., Drake, P.R.: PSO-based algorithm for home care worker scheduling in the uk. Computers & Industrial Engineering **53**(4) (2007) 559–583