Omega 40 (2012) 283–293

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

The falling tide algorithm: A new multi-objective approach for complex workforce scheduling

Jingpeng Li *, Edmund K. Burke, Tim Curtois, Sanja Petrovic, Rong Qu

School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, United Kingdom

ARTICLE INFO

Article history:
Received 23 June 2009
Accepted 8 May 2011
Processed by Associate Editor Smith
Available online 14 May 2011

Keywords:
Scheduling
Goal programming
Heuristics
Multi-criteria

ABSTRACT

We present a hybrid approach of goal programming and meta-heuristic search to find compromise solutions for a difficult employee scheduling problem, i.e. nurse rostering with many hard and soft constraints. By employing a goal programming model with different parameter settings in its objective function, we can easily obtain a coarse solution where only the system constraints (i.e. hard constraints) are satisfied and an ideal objective-value vector where each single goal (i.e. each soft constraint) reaches its optimal value. The coarse solution is generally unusable in practise, but it can act as an initial point for the subsequent meta-heuristic search to speed up the convergence. Also, the ideal objective-value vector is, of course, usually unachievable, but it can help a multi-criteria search method (i.e. compromise programming) to evaluate the fitness of obtained solutions more efficiently. By incorporating three distance metrics with changing weight vectors, we propose a new time-predefined meta-heuristic approach, which we call the falling tide algorithm, and apply it under a multi-objective framework to find various compromise solutions. By this approach, not only can we achieve a trade off between the computational time and the solution quality, but also we can achieve a trade off between the conflicting objectives to enable better decision-making.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Employee scheduling is an essential part of everyday business processes. A robust and powerful decision support system, which automates the scheduling of personnel has the potential to make significant savings in time and costs, as well as improving staff satisfaction. The challenge of building such a system has represented a series of demanding research questions, which have cut across operational research and artificial intelligence for over 40 years.

Nurse rostering represents a key challenge in healthcare personnel scheduling. Compared with some other types of employee scheduling problems, nurse rostering is often regarded as being more complicated due to the fact that hospitals operate 24 h a day and 7 days a week. This introduces more variables and extra constraints on night and weekend shifts. Of course, it is not the only problem with such characteristics but it still represents a particularly challenging problem where high quality solutions lead to high impact benefits. Also, unlike the rostering problems for many other service industries, nurse rostering needs to satisfy a range of different staff requirements (e.g. skills and shortage) on different days and shifts. In addition, when compared with many other rostering problems, nurse rostering usually has a longer planning horizon (e.g. one month) and more shift types (e.g. early, day, late and night shifts).

A survey was published in 2004, which presented a critical analysis of the important papers in the field of nurse scheduling [13]. Since then many more articles dealing with various nurse rostering problems have appeared in the literature [19]. It is clear that the field has generated significant international research interest in recent years.

The approaches in the literature range from traditional mathematical programming methods (e.g. [24]) to artificial intelligence methods such as constraint programming (e.g. [28]), expert systems [17] and knowledge based systems [5]. Particularly successful approaches to the problem are presented by meta-heuristic methods, which include genetic algorithms [21], simulated annealing [8], tabu search [11], memetic algorithms [2,10], variable neighbouhood searches [12] and component-based heuristics [27]. In very recent years, there has been an increasing level of interest in the study of mathematical programming based heuristics [3,6] and the study of hyper-heuristics for the problem [14]. However, most of the papers that have appeared in the literature have addressed simplified versions of the problem. The goal of developing methodologies that can handle the complexity of real world scenarios represents a major scientific challenge.

* Corresponding author. Tel.: +44 7879865914.
E-mail addresses: jpl@cs.nott.ac.uk (J. Li), ekb@cs.nott.ac.uk (E.K. Burke), tec@cs.nott.ac.uk (T. Curtois), sxp@cs.nott.ac.uk (S. Petrovic), rxq@cs.nott.ac.uk (R. Qu).

0305-0483/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.omega.2011.05.004

Nurse scheduling can be regarded as a type of resource allocation problem, in which the workload needs to be assigned to nurses periodically, taking into account a number of hard constraints (often generated by physical resource restrictions and legislation) and soft constraints (normally referred to as staff preferences). After obtaining a feasible solution which satisfies all the hard constraints, it is necessary to edit it for practical use by satisfying the soft constraints as much as possible. Hence, the nurse scheduling problem is inherently a multi-objective combinatorial problem, with each objective, possibly in conflict with other objectives, corresponding to a soft constraint.

In real situations, it is normally unrealistic to find a solution that satisfies all the soft constraints. Thus, it is possible to assign each constraint a weight and then seek a single optimal solution in the classical sense, by employing either goal programming or heuristic approach. However, the way of attaching a fix weight distribution is quite subjective and varies from person to person. A robust personnel scheduling system should be able to address the problem and to generate more alternatives for better decision-making. An efficient way for achieving such a task is to generate an approximate set of Pareto-optimal solutions automatically. The hospital administrator then imposes some general priority rules as a filter to significantly reduce the number of the schedules in the set. Those highly preferable remaining schedules are finally presented to the relevant nurses, who then select their favourite.

In this paper, we propose a new multi-objective approach that combines goal programming with a time-predefined search to solve real world nurse rostering problem instances in a Dutch hospital. The major difficulty of the problem lies in its large number of hard and soft constraints that reflect the real situation in a modern hospital. In the goal programming model, we formulate the hard constraints as system constraints and the soft constraints as individual goals. Due to the computational complexity, we do not use goal programming to solve the entire problem. Instead, we solve a partial problem to obtain two solutions: an ideal objective-value vector where each single objective under consideration achieves its optimal value and a coarse solution where only hard constraints are satisfied. The ideal objective-value vector acts as the reference point for a multi-criterion optimization technique called compromise programming [33] to efficiently evaluate the quality of obtained solutions, while the coarse solution acts as a starting point for the next meta-heuristic search to speed up the convergence.

We then propose a new global optimization algorithm, called the falling tide algorithm, to find an approximated Pareto set of the problem. It inherits the time-predefined feature from its predecessor, the great deluge algorithm [20], but has achieved a number of extensions that make its search more efficient and more effective.

This paper is organized in the following way. Section 2 presents a 0–1 goal programming model for the specific nurse rostering problem with many hard and soft constraints. Our multi-objective falling tide approach is described in Section 3. In Section 4, we discuss our results on 12 real-world data instances and in Section 5 we give some concluding remarks.

2. A goal programming model for nurse rostering with hard and soft constraints

The nurse rostering problem tackled here is derived from the real world situations of intensive care units in a Dutch hospital. It involves assigning four types of shifts (i.e. shifts of *early*, *day*, *late* and *night*) within a planning period of one month to 16 nurses of different working contracts in a ward. The problem has a number of constraints that can be categorized into two groups: hard constraints and soft constraints. Hard constraints are the ones that must be satisfied under any circumstances. Solutions that satisfy the hard constraints are called feasible solutions. Soft constraints are those that refer to desirable but not obligatory requirements. This problem can be formulated as a 0–1 goal programming model as follows.

2.1. Definition of parameters and variables

Goal programming is a branch of multi-objective search, which has been widely applied in multiple-criteria decision-making [31]. It can be thought of as an extension of linear programming to cope with multiple, normally conflicting objectives. To design our goal programming model, firstly we need to define the parameters and decision variables. Our definitions are outlined as follows.

Parameters:

I=set of nurses
J=set of days during the planning period
W=set index of weeks contained in the planning period
K=set of shift types represented as {1(*early*), 2(*day*), 3(*late*), 4(*night*)}

Decision variables:

x_{ijk}=1 if nurse i is assigned shift type k for day j, 0 otherwise

Additional parameters and variables will be introduced later in appropriate context.

2.2. Formulation of system constraints

System constraints of a goal programming model are the ones that can be expressed as linear equations or inequations that only consist of the decision variables. As hard constraints are the ones that must be satisfied absolutely, they can be treated as system constraints. Hence, we formulate each of the system constraints as follows.

Constraint 1: daily coverage demand (i.e. the number of nurses) of each shift type needs to be fulfilled.

$$\sum_{i \in I} x_{ijk} = d_{jk}, \quad \forall j \in \{1,\ldots,|J|\}, \ k \in K, \tag{1}$$

where d_{jk} is the coverage demand of type k on day j.

Constraint 2: a nurse should not work more than one shift on each day.

$$\sum_{k \in K} x_{ijk} \le 1, \quad \forall i \in I, \ j \in \{1,\ldots,|J|\}. \tag{2}$$

Constraint 3: a nurse must not work more than a certain number of working days during the planning period.

$$\sum_{j=1}^{|J|} \sum_{k \in K} x_{ijk} \le m_i, \quad \forall i \in I, \tag{3}$$

where m_i is the maximum number of working days for nurse i.

Constraint 4: a nurse must receive at least two weekends off duty during the planning period. Note that the first day in the planning period is a Monday, and so the 6th and 7th days are weekend days.

$$\sum_{w \in W} \sum_{k \in K} (x_{i(7w-1)k} + x_{i(7w)k}) \le 2|W| - 4, \quad \forall i \in I. \tag{4}$$

Constraint 5: a nurse must not work more than three *night* shifts during the planning period.

$$\sum_{j=1}^{|J|} x_{ij4} \le 3, \quad \forall i \in I. \tag{5}$$

Constraint 6: there is no *night* shift between two non-*night* shifts.

$$x_{i(j-1)4} - x_{ij4} + x_{i(j+1)4} \ge 0, \quad \forall i \in I, \quad j \in \{2,\ldots,|J|-1\}. \tag{6}$$

Constraint 7: there should be at least two holidays following a *night* shifts, i.e. no sequences of 'N01', 'N10' and 'N11', where 'N' denotes a *night* shift, '0' a holiday and '1' a working day.

$$x_{i(j-1)4} - \sum_{k=1}^{3} x_{ijk} + \sum_{k=1}^{3} x_{i(j+1)k} \le 1, \quad \forall i \in I,\ j \in \{2,\ldots,|J|-1\}, \tag{7}$$

$$x_{i(j-1)4} + \sum_{k=1}^{3} x_{ijk} - \sum_{k=1}^{3} x_{i(j+1)k} \le 1, \quad \forall i \in I,\ j \in \{2,\ldots,|J|-1\}, \tag{8}$$

$$x_{i(j-1)4} + \sum_{k=1}^{3} x_{ijk} + \sum_{k=1}^{3} x_{i(j+1)k} \le 2, \quad \forall i \in I,\ j \in \{2,\ldots,|J|-1\}. \tag{9}$$

Constraint 8: there is an upper limit for the number of consecutive *night* shifts of each nurse.

$$\sum_{j=r}^{r+n_1} x_{ij4} \le n_1, \quad \forall i \in I,\ r \in \{1,\ldots,|J|-n_1\}, \tag{10}$$

where n_1 is such an upper limit value.

Constraint 9: there is an upper limit for the number of consecutive working days of each nurse.

$$\sum_{j=r}^{r+n_2} \sum_{k \in K} x_{ijk} \le n_2, \quad \forall i \in I,\ r \in \{1,\ldots,|J|-n_2\}, \tag{11}$$

where n_2 is such an upper limit value.

Constraint 10: a particular nurse i_1 cannot work *late* shifts.

$$x_{i_1(j)3} = 0, \quad \forall j \in \{1,\ldots,|J|\}. \tag{12}$$

2.3. Formulation of goals

Goal programming is an extension of the linear programming technique that treats some constraints of the linear programming problem as the goals. To each goal, slack or surplus variables are introduced to represent the positive or negative deviations from the goal. Hence, we can formulate the goals, with each goal corresponding to a soft constraint, as follows.

Goal 1: This attempts to establish complete weekends, i.e. it aims to have either no shifts or two shifts in weekends, during the planning period. It can be represented as

$$\sum_{k \in K} (x_{i(7w-1)k} - x_{i(7w)k}) - d1_{iw} = 0, \quad \forall i \in I,\ w \in W, \tag{13}$$

where $d1_{iw}$ is the amount of deviation from goal 1 for nurse i working in week w.

Goal 2: This tries to avoid any stand-alone shift, i.e. a working day between two free days, during the planning period. It can be formally specified as follows:

$$\sum_{k \in K} (x_{i(j-1)k} - x_{ijk} + x_{i(j+1)k}) + d2_{ij}^{-} \ge 0, \quad \forall i \in I,\ j \in \{2,\ldots,|J|-1\}, \tag{14}$$

where $d2_{ij}^{-}$ is the amount of negative deviation from goal 2 for nurse i working on day j.

Goal 3: This attempts to allocate a minimum of two consecutive holidays at a time. It can be formally outlined as follows:

$$\sum_{k \in K} (x_{i(j-1)k} - x_{ijk} + x_{i(j+1)k}) - d3_{ij}^{+} \le 1, \quad \forall i \in I,\ j \in \{2,\ldots,|J|-1\}, \tag{15}$$

where $d3_{ij}^{+}$ is the amount of positive deviation from goal 3 for nurse i working on day j.

Goal 4: This attempts to allocate no more than a certain number of consecutive days on a particular shift type. It can be formally represented as

$$\sum_{j=r}^{r+3} x_{ijk} - d4_{irk}^{+} \le c_k, \quad \forall i \in I,\ r \in \{1,\ldots,|J|-3\},\ k \in \{1,3\}, \tag{16}$$

where c_k is the maximum number of consecutive shifts of type k, and $d4_{irk}^{+}$ is the amount of positive deviation from goal 4 for nurse i of shift type k working on day j.

Goal 5: This aims to allocate no less than two consecutive shifts of a particular shift type (i.e. early and late) during the planning period. It can be formally represented as follows:

$$x_{i(j-1)k} - x_{ijk} + x_{i(j+1)k} + d5_{ijk}^{-} \ge 0, \quad \forall i \in I,\ j \in \{2,\ldots,|J|-1\},\ k \in \{1,3\}, \tag{17}$$

where $d5_{irk}^{-}$ is the amount of negative deviation from goal 5 for nurse i of shift type k working on day j.

Goal 6: This attempts to allocate no more than a certain number of weekly working days to each nurse with a specific type of working contract. A formal representation is

$$\sum_{j=7w-6}^{7w} \sum_{k \in K} x_{ijk} - d6_{tiw}^{+} \le g_t, \quad \forall t \in \{1,2,3\},\ i \in I_t,\ w \in W, \tag{18}$$

where I_t is the subset of nurses working on the tth contract satisfying $I=\{I_1$ (full time), I_2 (short part time), I_3 (long part time)$\}$, g_t is the maximum number of weekly working days of nurses in subset I_t, and $d6_{tiw}^{+}$ is the amount of positive deviation from goal 6 for nurse i of contract type t working in week w.

Goal 7: This aims to allocate no less than a certain number of weekly working days to each nurse with different working contract. It can be formally outlined as

$$\sum_{j=7w-6}^{7w} \sum_{k \in K} x_{ijk} + d7_{tiw}^{-} \ge h_t, \quad \forall t \in \{1,2,3\},\ i \in I_t,\ w \in W, \tag{19}$$

where h_t is the minimum number of weekly working days of nurses in subset I_t, and $d7_{tiw}^{-}$ is the amount of positive deviation from goal 7 for nurse i of contract type t working in week w.

Goal 8: This attempts to allocate a maximum of three consecutive working days for part-time nurses during the planning period. A formal representation is

$$\sum_{j=r}^{r+3} \sum_{k \in K} x_{ijk} - d8_{ir}^{-} \le 3, \quad \forall i \in I_1,\ r \in \{1,\ldots,|J|-3\}, \tag{20}$$

where $d8_{ir}^{-}$ is the amount of negative deviation from goal 8 for nurse i working on day r.

Goal 9: This avoids certain shift type successions, e.g. a *day* shift followed by an *early* shift, during the planning period. It can be formally represented as follows:

$$x_{ijk_1} + x_{i(j+1)k_2} - d9_{ijk}^{-} \le 1, \quad \forall i \in I,\ j \in \{1,\ldots,|J|-1\},\ (k_1,k_2) \in K', \tag{21}$$

where K' is the set of undesirable shift type successions represented as $\{(2,1), (3,1), (3,2), (1,4)\}$, and $d9_{ijk}^{-}$ is the amount of negative deviation from goal 9 for nurse i of shift type k working on day j.

Table 1
A breakdown of the total number of variables and constraints for a typical problem.

A typical problem ($\|I\|=16$, $\|J\|=35$, $\|K\|=4$)	Number of variables	Number of constraints
System constraints	2257	3816
Goal 1	80	80
Goal 2	528	528
Goal 3	528	528
Goal 4	1040	1040
Goal 5	819	819
Goal 6	67	67
Goal 7	64	64
Goal 8	1120	1120
Goal 9	2192	2192
Total	8695	10,254

2.4. Objective function

The objective of goal programming is to minimize the sum of the weighted deviations from individual goals. For each goal, there is a weight that reflects the relative importance of this goal compared to the others. Let w_t be the weight of goal t. The objective function of our problem can be formulated as

$$\begin{aligned}\text{Minimize}\quad Z = {} & w_1\sum_{i\in I}\sum_{w\in W}(d1_{iw}) + w_2\sum_{i\in I}\sum_{j=2}^{|J|-1} d2^-_{ij} + w_3\sum_{i\in I}\sum_{j=2}^{|J|-1} d3^+_{ij} \\ & + w_4\sum_{i\in I}\sum_{r=1}^{|J|-3}\sum_{k\in\{1,3\}} d4^+_{irk} + w_5\sum_{i\in I}\sum_{j=2}^{|J|-1}\sum_{k\in\{1,3\}} d5^-_{ijk} \\ & + w_6\sum_{t\in\{1,2,3\}}\sum_{i\in I_t}\sum_{w\in W} d6^+_{tiw} + w_7\sum_{t\in\{1,2,3\}}\sum_{i\in I_t}\sum_{w\in W} d7^-_{tiw} \\ & + w_8\sum_{i\in I_1}\sum_{r=1}^{|J|-3} d8^-_{ir} + w_9\sum_{i\in I}\sum_{j=2}^{|J|-1}\sum_{k\in K'} d9^+_{ijk}. \end{aligned} \tag{22}$$

We have attempted to solve the above goal programming problem by employing ILOG CPLEX 10.0. However, compared with the results of other approaches on the set of benchmark problems that we study, the results of goal programming are unsatisfactory even if we allow an additional overnight runtime for each problem. Studying a typical problem of scheduling 16 nurses of 4 shift types during a 5-week period, we find that the computational complexity is mostly brought about by the goals as they comprise 74% of the variables and 63% of the constraints (see Table 1). Hence, to better handle these goals, we need to seek a new approach that employs a simplified model of goal programming but is hybridized with an extra (meta-)heuristic search. In addition, unlike a classical goal programming method that produces a single final solution towards a user's specific preference setting, the new approach should be able to produce a set of non-dominated solutions in one go for the decision maker's evaluation. A number of multi-objective approaches have been developed to determine such a set [1,4,22,27,32,34].

3. A time-predefined multi-objective approach to nurse scheduling

In this section, we present an alternative time-predefined approach to deal with the multi-objective nurse scheduling problem.

3.1. Objective functions of individual goals

We formulate the objectives of our meta-heuristic model as function $f_t(x)$, $t=1,\ldots,9$, where x denotes a decision variable vector of $|I|\times|J|\times|K|$ and f_t corresponds to the tth goal described in Section 2.3 as follows:

$$\text{Minimize}\quad f_1(x) = \sum_{i\in I}\sum_{w\in W}\left|\sum_{k\in K}(x_{i(7w-1)k} - x_{i(7w)k})\right|, \tag{23}$$

$$\text{Minimize}\quad f_2(x) = \sum_{i\in I}\sum_{j=2}^{|J|-1}\max\left\{0, \sum_{k\in K}(-x_{i(j-1)k} + x_{ijk} - x_{i(j+1)k})\right\}, \tag{24}$$

$$\text{Minimize}\quad f_3(x) = \sum_{i\in I}\sum_{j=2}^{|J|-1}\max\left\{0, \sum_{k\in K}(x_{i(j-1)k} - x_{ijk} + x_{i(j+1)k}) - 1\right\}, \tag{25}$$

$$\text{Minimize}\quad f_4(x) = \sum_{i\in I}\sum_{r=1}^{|J|-3}\sum_{k\in\{1,3\}}\max\left\{0, \sum_{j=r}^{r+3} x_{ijk} - c_k\right\}, \tag{26}$$

$$\text{Minimize}\quad f_5(x) = \sum_{i\in I}\sum_{j=2}^{|J|-1}\sum_{k\in\{1,3\}}\max\{0, -x_{i(j-1)k} + x_{ijk} - x_{i(j+1)k}\}, \tag{27}$$

$$\text{Minimize}\quad f_6(x) = \sum_{t=1}^{3}\sum_{i\in I_t}\sum_{w=1}^{|W|}\left[\max\left\{0, \sum_{j=7w-6}^{7w}\sum_{k\in K} x_{ijk} - g_t\right\}\right], \tag{28}$$

$$\text{Minimize}\quad f_7(x) = \sum_{t=1}^{3}\sum_{i\in I_t}\sum_{w=1}^{|W|}\left[\max\left\{0, h_t - \sum_{j=7w-6}^{7w}\sum_{k\in K} x_{ijk}\right\}\right], \tag{29}$$

$$\text{Minimize}\quad f_8(x) = \sum_{i\in I_1}\sum_{r=1}^{|J|-3}\max\left\{0, \sum_{j=r}^{r+3}\sum_{k\in K} x_{ijk} - 3\right\}, \tag{30}$$

$$\text{Minimize}\quad f_9(x) = \sum_{i\in I}\sum_{j=1}^{|J|-1}\sum_{(k_1,k_2)\in K'}\max\{0, x_{ijk_1} + x_{i(j+1)k_2} - 1\}. \tag{31}$$

3.2. The Pareto set

With the above nine goals, we regard the nurse scheduling problem as a multi-objective optimization problem represented as

$$\text{Minimize}\quad f(x) = (f_1(x),\ldots,f_9(x))^T, \tag{32}$$

subject to $x\in S$, where x represents a decision variable vector and S is the set of candidate solutions defined by the hard constraints 1–10.

In real world nurse rostering with many hard and soft constraints, due to the conflicting nature of some objectives, it is impossible to find a solution at which the objective functions would achieve their minimal values simultaneously, and thus, the classical concept of a commonly optimal solution does not apply. Under this circumstance, the concept of a Pareto-optimal solution is explained. When comparing two solutions for a minimization problem, a solution x is said to dominate another solution y if $\exists t\in\{1,\ldots,9\}|f_t(x)<f_t(y)$, and $\forall t'\in\{1,\ldots,9\}$, and $t'\neq t|f_{t'}(x)\leq f_{t'}(y)$. A solution is called *a Pareto-optimal solution* of the problem if there are no other solutions that can dominate it, or in other words, it represents an optimal trade off between conflicting objectives. The set of all Pareto-optimal solutions is called *the*

Pareto set. The image $f(x^*)$ of a Pareto-optimal solution x^* is called *the efficient solution*, and the set of all the efficient solutions is called *the Pareto front*. Note in this paper, as we apply meta-heuristic search to address the problem, the Pareto set we generate can only be regarded as an approximation to the true Pareto set.

3.3. Fitness evaluation by compromise programming

A number of approaches have been developed to search for the Pareto set of a multi-objective problem, among which a class of approaches uses a value function to aggregate multiple objectives into one. Due to the existence of conflicting objectives, properly expressing the user's preferences by a single-objective function is difficult. Solutions produced by such methods must not be dominated by any other solution and must reconcile the various conflicting objectives to represent an appropriate expression of the user's preferences. Such solutions are regarded as compromise solutions.

Compromise programming is a multiple-criteria decision-making technique initially proposed by Zeleny [32] to obtain compromise solutions. Its basic idea is firstly to identify an ideal solution as a point where each single objective under consideration achieves its optimal value, and then the approach seeks a solution that is as close to the ideal point as possible. For our nurse rostering problem, we apply the above 0–1 goal programming solver 9 times, once for each goal (or objective) without the consideration of other goals. Hence, what we have after solving the problem 9 times is an ideal objective-value vector, that is $v=(f_1(u_1), f_1(u_2), \ldots, f_9(u_9))$ where u_t is the optimal solution with respect to the objective t. That can then be used to measure the deviation of candidate solutions. With the power of CPLEX, the optimum value of each objective (i.e. $f_t(u_t)$) can be easily found as the solution space is significantly reduced (see Table 1).

After generating an ideal objective-value vector v that is generally unreachable in practise, we apply compromise programming with the L_p-metric to measure the fitness of a current solution x (i.e. we calculate the weighted distance between v and x) as

$$L_p(x)=\left\{\sum_{t=1}^{9}[w_t(f_t(x)-f_t(u_t))]^p\right\}^{1/p}, \tag{33}$$

where $w=(w_1,\ldots,w_9 \,|\, w_t>0,\ t=1,\ldots,9)^T$ is a weight vector with each w_i corresponding to the importance of a particular objective, and $p\in\{1,2,\ldots\}\cup\{\infty\}$ defines the type of metric.

The parameter w_t characterizes a users' preference. It represents the relative importance of one objective against another. Simply stated, it places emphasis on the objective that the user deems to be important. The parameter is needed because different users in the decision-making process would have different viewpoints concerning the important objectives.

The parameter p represents the importance of the maximal deviation from the ideal point. Typically, as p increases, the weighting of the deviations also increases. Varying the parameter from 1 to infinity enables one to move from having a high level of compensation among the objectives to having no compensation at all. Three values of parameter p are of particular importance: $p=1,2,\infty$. When $p=1$, the definition of $L_1(x)=\sum_{t=1}^{9}[w_t(f_t(x)-f_t(u_t))]$ yields the so-called Manhattan metric, by which all deviations are weighted equally. When $p=2$, the definition of

$$L_2(x)=\sqrt{\sum_{t=1}^{9}[w_t(f_t(x)-f_t(u_t))]^2}$$

yields the so-called Euclidean metric, by which the deviations are weighted in proportion to their magnitude. When $p=\infty$, the definition of $L_\infty=\max_{t=1,\ldots,9}\{w_t[f_t(x)-f_t(u_t)]\}$ yields the so-called Tchebycheff metric, by which only the maximal deviation is considered.

Compromise programming with L_1 is equivalent to the weighted-sum method, which works very well for convex problems, i.e. problems whose objective functions are convex and whose solution spaces are convex sets. Geoffrion [23] shows that for every efficient solution of a convex problem there exists a positive weight vector. However, there is a problem for the L_1 metric to deal with concave problems as there may not exist a weight vector such that a given efficient solution can be found. To help the understanding, we use Fig. 1 to illustrate the Pareto front of a bi-objective problem in the objective space. The efficient solutions found by the L_1 metric can be geometrically identified as the contacting points between the curve and the line supporting the curve and perpendicular to the weight vector w. From Fig. 1, we can see that the L_1 metric fails to generate the efficient solutions located on the arc between contact points A and B, as for some $w>0$, it always achieves a better (smaller) weighted-sum value by supporting the Pareto curve outside of the arc rather than at any point along the arc.

On the other hand, compromise programming with L_α is very useful in generating Pareto solutions for both convex and concave problems. Bowman [7] shows that for every Pareto-optimal solution there exists a positive weight vector. Fig. 2 shows the same Pareto front illustrated in Fig. 1. For a given ideal point v and a weight vector w, the efficient solutions found by the L_α metric can be geometrically identified as the contacting points between the Pareto front and the corresponding iso-value curve of L_α. Hence, by keeping the point v the same but changing the vector w, we may achieve all the efficient points located on the arc between points A and B.

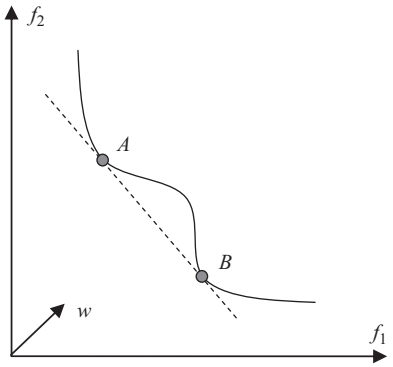

Fig. 1. The generation of efficient solutions by the L_1 metric.

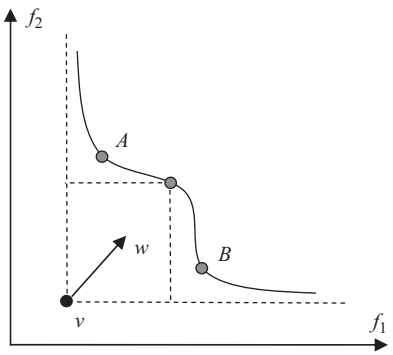

Fig. 2. The generation of efficient solutions by the L_α metric.

Hence, in order to effectively generate an approximated Pareto set for our nurse rostering problem, we should incorporate both metrics L_1 and L_α in measuring the fitness of obtained solutions. In addition, in order to achieve a better compensation among the objectives, we should incorporate at least one nonlinear metric of L_p where $p \in [2, \infty)$. Due to its simplicity, in this paper we only consider the situation of $p=2$.

3.4. A falling tide algorithm for multi-objective optimization

The great deluge algorithm is presented in [20]. It is so called because it can be illustrated by visualizing a solution landscape which is subject to a "deluge" which "forms" the search mechanism towards potential solutions. In this section, we propose a new approach which is inspired by the great deluge algorithm and which we have called the falling tide algorithm. This basic idea can be illustrated by considering a hypothetical situation where someone is collecting shells on a beach during a falling tide period and is wanting to stay out of the water. The person moves by a random walk on the beach while the tide is falling. During this period, the sea level will become lower and lower. However, big waves will hit the beach causing the sea level to temporarily rise again. The wave could bring on new shells from the ocean and take them back into the sea with it. The person stays as close to the sea as possible (moving back and forth with the waves). When the falling tide ceases or the time has run out, the person stops collecting shells. If the person is unhappy with the gains, then it would be possible to come back again during the next falling tide period.

More technically, the steps of our basic falling tide algorithm can be summarized as follows:

1. Let N_{run} be the number of runs (i.e. the number of tides), N_{wav} the number of waves defined in each run, and N_{lev} the number of levels defined in each wave regression.
2. Set the initial solution x_0 and calculate its cost by function $f(x_0)$.
3. Set the current solution x to be x_0. Further set the initial level $B=f(x)/\alpha(x)$, where the restart coefficient $\alpha(x)$ (i.e. a random variable corresponding to the child's arriving time or searching region) is generated as a random number between 0 and 1.
4. Set the regression rate $\Delta B=(B-f(u))/N_{lev}$, where u represents the ideal solution.
5. Define the neighbourhood $N(x)$ and randomly select a candidate solution $x^* \in N(x)$.
6. Calculate $f(x^*)$. If $f(x^*) \le B$ or $f(x^*) \le f(x)$, then accept x^* as x.
7. Lower the level $B=B-\Delta B$.
8. Repeat steps 5–7 until the number of N_{lev} levels is carried out.
9. Reset $B=(\beta(x^*)+1)f(x^*))$, where the water-rerising rate $\beta(x^*)$ (i.e. a random variable corresponding to the unpredictable wind direction) is generated as a random number between 0 and 1.
10. Repeat steps 4–9 until the number of N_{wav} waves is carried out.
11. Repeat steps 3–10 until the number of N_{run} runs is carried out.
12. Output the best solution found.

As mentioned above, our falling tide algorithm can be viewed as an extension to the great deluge algorithm, which was first proposed by Dueck as an alternative to simulated annealing. Like the great deluge algorithm, a falling tide algorithm may accept worse solutions during its run if their cost values are no larger than a level value, which is lowered at each iteration by a given rate. However, our falling tide approach has achieved the following three distinct improvements that make its search more efficient and more effective.

Firstly, taken as a whole, the granularity of level reduction with the falling tide approach will change adaptively and tend to be finer and finer after more and more waves, although the "level" reduction within each wave is still kept at a fixed rate. This characteristic may result in a quicker convergence within the solution space.

Secondly, the great deluge algorithm lacks the ability to jump out of local optima, although it can accept worse solutions and has achieved certain success in a wide variety of areas (e.g. exam timetabling [9,31] and channel assignment [25]). The drawback is caused by the fact that, at each of its iterations, the region containing the solutions with cost values above the current level (i.e. those under the current wave level in the "deluge" analogy) can never be explored. Burke et al. [9] revised the great deluge algorithm by accepting all downhill moves during the search, as well as making an additional local improvement after its search. Superior results were reported on some benchmark problems due to the larger region explored and the longer execution time allowed, but unfortunately those revisions do not alter the fact that the method is vulnerable to converging too quickly. For a falling tide algorithm, the situation is alleviated by its scheme of regularly raising the water level when a new wave regression starts.

Thirdly, for an algorithm that only uses a "rising" water scheme, solutions with sufficiently large cost increases will still not be accepted. However, these larger increases may be required in order for the algorithm to proceed to a global optimum. Thus, no matter how small the water decay rate is and how often the waves are raised, from a certain point on, the algorithm may be "closed off" from finding a global optimum. In our falling tide algorithm, this issue is addressed by occasionally restarting the search and using the same initial solution but with a different initial level value generated at random.

The major advantage of the great deluge algorithm lies in its *time-predefined* feature, that is, throughout the search there is only one controlling parameter of N_{lev} (i.e. the number of levels). Being an extension of the great deluge algorithm, our falling tide algorithm inherits this feature by introducing two more time-related parameters: N_{run} (i.e. the number of runs) and N_{wav} (i.e. the number of waves). As the CPU time consumed in each level on the same machine is roughly the same, this feature provides users with a straightforward way to define a certain amount of time in which the algorithm should run: the larger these three parameter values, the better the solution quality will be. Based on this consideration, we apply the falling tide algorithm under a multi-objective framework to solve our nurse rostering problem as follows:

1. Let N_{run} be the number of runs, N_{wav} the number of waves defined in each run, and N_{lev} the number of levels defined in each wave regression.
2. Apply the goal programming model (described in Section 2) with different settings of p_t values in function (22) to obtain an initial solution x_0 and an ideal objective-value vector $v=(f_1(u_1), f_1(u_2), \ldots, f_9(u_9))$: x_0 is obtained by setting $p_t=0 \,|\, \forall t \in \{1,\ldots,9\}$, while v is obtained by running a goal programming solver nine times and at the tth time setting $p_{t'}=0 \,|\, \forall t' \in \{1,\ldots,9\}, t' \neq t$.
3. Let D be the set of potentially non-dominated solutions, set $D=\{x_0\}$.
4. Randomly express user's preference on each objective (i.e. the weight vector $w=(w_1,\ldots,w_9)^T$), and randomly select an L_p metric where $p \in \{1,2,\infty\}$.
5. Set the current solution x to be x_0, calculate the fitness of x as $L_p(x)=\{\sum_{t=1}^{9}[w_t(f_t(x)-f_t(u_t))]^p\}^{1/p}$, and define the initial level

as $B=L_p(x)/\alpha(x)$, where the restart coefficient $\alpha(x)$ is a random number between 0 and 1.
6. Set the regression rate $\Delta B=L_p(x)/N_{lev}$.
7. Construct a new solution y, $y \in N(x)$, in the following way: first generate a random number n between one and the length of the planning period, then randomly locate n days (which need not represent a continuous sequence), and then vertically swap the shifts of those n days between a randomly selected pair of nurses in solution x.
8. If y violates at least one of the hard constraints, go to Step 7.
9. Calculate the fitness of y as $L_p(y)=\{\sum_{t=1}^{9}[w_t(f_t(y)-f_t(u_t))]^p\}^{1/p}$.
10. If $L_p(y) \le L_p(x)$ or $L_p(y) \le B$, accept y and replace x with y.
11. If y is accepted, update set D with y in the following way: check y for Pareto dominance among all the solutions in D, add y to D if it is non-dominated, and remove the solutions originally in D that are dominated by y.
12. Lower the level $B=B-\Delta B$.
13. Repeat steps 7–12 until the number of N_{lev} levels is carried out.
14. Reset $B=(\beta(y)+1)f(y)$, where the water-rerising rate $\beta(y)$ is a random number generated between 0 and 1.
15. Repeat steps 6–14 until the number of N_{wav} waves is carried out.
16. Repeat steps 4–15 until the number of N_{run} runs is carried out.
17. Output set D as an approximation of the Pareto-optimal set.

Generally, two strategies are very important in the design of a meta-heuristic algorithm: they are diversification (i.e. driving the search into new promising regions) and intensification (i.e. focusing on the search into attractive regions). Our falling tide algorithm reflects these two strategies in the following three ways.

Firstly, within each wave regression, at the earlier stage when the current water level is high, the diversification mechanism is mainly performed by accepting most of the generated solutions. At later stages when the current water level becomes lower and lower, the intensification mechanism is mainly performed by only accepting the small-perturbed solutions.

Secondly, within each falling tide, a number of waves are defined, and the search between two consecutive waves focuses more on the intensification aspect, although the search within a single wave regression still implements gradually from diversification to intensification. In general, the search from a new wave within the same tide can be regarded as a continuation of the search from the previous wave, because the initial level of the new wave is mainly determined by the current solution and the effect of a random variable β (corresponding to the unpredictable wind direction) is minor.

Thirdly, for an entire run of our falling tide algorithm, a number of tides are defined, and the search between two consecutive tides focuses more on the diversification aspect. For each new tide, its initial search point is reshuffled because the initial water level of the first wave regression in the tide is mainly determined by another random variable α that corresponds to the starting search region.

4. Computational results

In this section, we present the results of extensive experiments on 12 real data instances, with each instance corresponding to a calendar month from January to December in 2003. These instances are provided by ORTEC, an international consultancy company specializing in planning, optimization, and decision support solutions. We first compare our approach with others in the literature by using the same weighted-sum objective function. Then, we present the results under the multi-objective framework.

4.1. Results under a traditional weighted sum objective function

Table 2 lists the benchmark results of three previous approaches. They are a hybrid genetic algorithm [29] denoted as "Hybrid GA", a hybrid variable neighbourhood search denoted as "Hybrid VNS", and an integer programming based variable neighbourhood search [16] denoted as "IP-based VNS". These three approaches use the same weighted-sum objective function to aggregate all the objectives, which is:

$$\text{Minimize } f(x)=\sum_{t=1}^{9} w_t f_t(x), \tag{34}$$

where vector w=[1000,1000,100,10,10,10,10,10,5]. These values are set based on the following priority ordering after consultation with the hospital: [goals 1–2]$\succ$[goal 3]$\succ$[goals 4–8]$\succ$[goal 9], where '$\succ$' denotes "*more preferred than*".

The hybrid GA and the hybrid VNS were coded in Delphi 5 and run on a Pentium 1.7 GHz PC under Windows 2000. The IP-based VNS was run on a 2.0 GHz PC under Windows XP, of which the IP part was solved by CPLEX 10.0 and the VNS part was coded in Java 2. In general, the IP-based VNS has produced better results than the other two approaches.

Our proposed falling tide algorithm is implemented under the same environment as the IP-based VNS. For each data instance, we set the runtime (corresponding to a fixed number

Table 2
Costs of previous approaches after 1-h runtime.

Data	Hybrid GA	Hybrid VNS	IP-based VNS
Jan	775	735	460
Feb	1791	1866	1526
Mar	2030	2010	1713
Apr	612	457	391
May	2296	2161	2090
Jun	9466	9291	8826
Jul	781	481	425
Aug	4850	4880	3488
Sept	615	647	330
Oct	736	665	445
Nov	2126	2030	1613
Dec	625	520	405
Ave.*	**2225**	**2145**	**1809**

Ave.*: the average value.

Table 3a
Results of 10 independent runs under a weighted-sum objective function.

Data	Initial solution		60 s ($N_{run}=10$, $N_{wav}=10$, $N_{lev}=4\times10^5$)				
	Cost	Time (s)	Min	Δ^* (%)	Mean	Δ^* (%)	Dev
Jan	29,025	16	410	10.9	504.2	−9.6	51.6
Feb	17,467	11	1520	0.4	2424.4	−58.9	978.3
Mar	40,558	12	1530	10.7	1596.7	6.8	53.8
Apr	43,360	15	280	28.4	794.3	−52.0	794.4
May	39,627	21	870	58.4	1338.8	35.9	795.5
Jun	44,737	26	8906	−0.9	9068.7	−2.7	97.0
Jul	30,376	16	290	31.8	744.0	−75.1	787.9
Aug	36,911	17	1345	61.4	1803.8	48.3	796.5
Sep	38,791	20	210	36.4	464.8	−40.8	580.2
Oct	41,127	21	485	−9.0	744.1	−67.2	592.8
Nov	34,238	22	1560	3.3	1623.3	−0.6	49.0
Dec	35,501	19	190	53.1	626.8	−54.8	782.9
Ave.*	**35,977**	**18**	**1466**	**19.0**	**1811.2**	**0.8**	**530.0**

Δ^*: relative percentage deviation from the previously best known solution; Ave.*: the average value.

Table 3b
Results of 10 independent runs under a weighted-sum objective function.

Data	300 s ($N_{run}=10$, $N_{wav}=10$, $N_{lev}=2\times10^6$)					1200 s ($N_{run}=10$, $N_{wav}=20$, $N_{lev}=4\times10^6$)				
	Min	Δ^* (%)	Mean	Δ^* (%)	Dev	Min	Δ^* (%)	Mean	Δ^*	Dev
Jan	405	12.0	440.8	4.2	16.7	335	27.2	381.6	17.0	26.5
Feb	1455	4.7	1524.2	0.1	34.0	1426	6.6	1505.3	1.4	50.9
Mar	1440	15.9	1540.9	10.0	46.8	1435	16.2	1488.0	13.1	26.9
Apr	240	38.6	301.3	22.9	55.2	181	53.7	232.4	40.6	51.6
May	790	62.2	1264.6	39.5	822.5	756	63.8	807.8	61.3	22.9
Jun	8965	−1.6	9053.3	−2.6	63.5	8890	−0.7	8962.2	−1.5	49.1
Jul	210	50.6	461.6	−8.6	603.0	160	62.4	238.7	43.8	41.9
Aug	1265	63.7	1507.0	56.8	594.0	1225	64.9	1259.3	63.9	32.1
Sept	90	72.7	538.7	−63.2	798.2	80	75.8	109.6	66.8	22.3
Oct	395	11.2	833.4	−87.3	802.3	345	22.5	389.4	12.5	21.3
Nov	1486	7.9	1561.4	3.2	44.6	1456	9.7	1505.1	6.7	24.6
Dec	92	77.3	334.9	17.3	611.2	65	84.0	95.2	76.5	17.6
Ave.*	**1403**	**22.4**	**1613.5**	**10.8**	**374.3**	**1363**	**24.7**	**1414.6**	**21.8**	**32.3**

Δ^*: relative percentage deviation over the previously best known solution; Ave.*: the average value.

of iterations) after the initial solution to be 60, 300, and 1200 s. Each resulting solution is evaluated by the same function given in equation (34). To test the robustness of our approach, each instance was run ten times with different pseudo random seeds.

Table 3a lists the solution cost of each initial solution and the CPU time for CPLEX 10.0 to achieve it. Table 3b lists the summary results of 10 runs for each group of experiments with a different runtime. To study the distribution of results, the indices of min, mean, and standard deviation are applied.

The computational results in Tables 3a and 3b reveal that the performance of our approach is as expected: the longer the runtime, the better the solution quality and the smaller the variation of values between the runs. Compared with the IP-based VNS, which was the previous best performing method, our new approach has achieved significantly better results on 11 out of the 12 instances within much shorter runtime. In terms of the best results and the average results, on average our approach improves the previous best results (obtained after 1 hour's runtime) by 19.0% and 0.8% after 60 s, 22.4% and 10.8% after 300 s, and 24.7% and 21.8% after 1200 s.

Fig. 3 depicts the searching process employed by the great deluge algorithm and the falling tide algorithm (with $N_{run}=3$, $N_{wav}=4$, and $N_{lev}=7000$) for the Mar instance. The x-axis represents the number of iterations (with each iteration corresponding to one change of the level value), and the y-axis represents the solution cost. "Falling Tide-1" corresponds to the first run of the falling tide algorithm while $\alpha(x)=1.0$ and $\beta(x)=0$, "Falling Tide-2" represents the second run of the falling tide algorithm while $\alpha(x)=0.9$ and $\beta(x)=0.5$, and "Falling Tide-3" describes the third run of the falling tide algorithm while $\alpha(x)=0.8$ and $\beta(x)=1.0$. This figure clearly shows that search by our falling tide algorithm is more efficient and more effective. Regarding the other instances, the characteristic shapes of the curves are very similar although their actual values may differ.

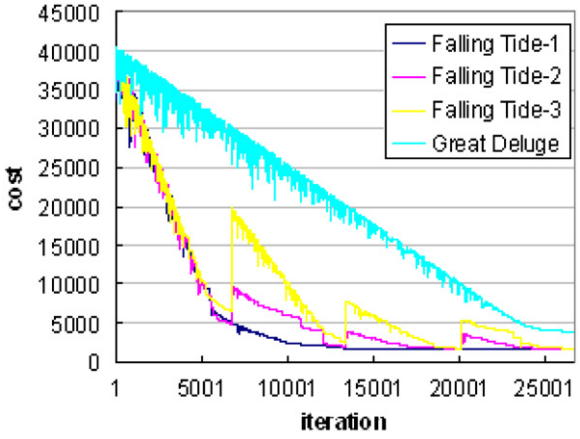

Fig. 3. Comparison of a great deluge algorithm and our falling tide algorithm (with $N_{run}=3$ for the Mar instance).

4.2. Results under a multi-objective framework

For multi-objective problems, there are various performance metrics to measure individual algorithms in terms of convergence and diversity, among which the four metrics of error ratio, generational distance, spacing and two set coverage are commonly used. The error ratio metric indicates the percentage of solutions (from the non-dominated vectors found) that are not members of the true Pareto set. The generational distance metric introduces a way of estimating how far the elements in the Pareto front produced by an algorithm are from those in the true Pareto front. The spacing metric proposes a way of measuring the distance variance of neighbouring vectors in the known Pareto front. The two set coverage metric defines the percentage of the solutions in one approximation of the Pareto front that are dominated by at least one solution in another approximation of the Pareto front.

The first three metrics need to know the true Pareto set/front in advance, which is unachievable in our situation as no existing method can solve such a nine-objective large size problem to Pareto optimality. Thus, none of them can be applied to assess the performance of our proposed approach under a multi-objective framework. With regard to the last metric, it requires the approximated Pareto set produced by another approach, and paper [15] is the only one in the literature that tackles the same problem by a different multi-objective method. Hence, we use the two set coverage metric to compare the results of those two approaches.

Assume A denotes the approximated Pareto set generated by the falling tide algorithm, and B denotes the approximated Pareto set reported in [15]. The two set coverage metric maps the ordered pair of sets (A,B) to the interval [0,1] using the following equation (34):

$$C(A,B)=\frac{|\{b\in B;\ \exists a\in A: a\geq b\}|}{|B|}. \tag{35}$$

Table 4
Results of a multi-objective comparison.

Data	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sept	Oct	Nov	Dec
$C(A,B)$	0.85	0.71	0.80	0.92	0.86	0.74	0.93	0.84	0.90	0.76	0.78	0.94

Table 5
Results of multi-objective optimization.

Data	Initial solution		100 runs with different w_i and p values in L_p metrics (60 s per run)						200 runs with different w_i and p values in L_p metrics (60 s per run)					
	Vio	%	$\|D\|$	Min	%	Mean	%	Dev	$\|D\|$	Min	%	Mean	%	Dev
Jan	518	5.6	543	50	0.5	115.8	0.5	68.8	500	42	0.5	76.5	0.8	19.4
Feb	285	3.4	1190	160	1.9	227.8	1.9	52.8	1325	156	1.8	245.4	2.9	54.4
Mar	716	7.9	1125	159	1.8	254.8	1.8	62.6	1093	154	1.7	237.4	2.6	53.8
Apr	578	6.6	544	34	0.4	172.6	1.3	210.5	321	29	0.3	49.1	0.6	9.2
May	669	7.3	1029	89	1.0	135.8	2.5	22.6	878	87	0.9	130.7	1.4	25.4
Jun	337	3.8	686	201	2.3	244.0	2.9	38.1	631	196	2.2	229.2	2.6	21.5
Jul	495	5.5	540	31	0.3	173.7	1.9	190.7	354	23	0.3	46.5	0.5	11.3
Aug	670	7.2	1819	133	1.4	198.4	1.5	34.3	1625	128	1.4	202.1	2.2	43.9
Sept	546	6.3	279	17	0.2	40.6	2.8	9.5	242	17	0.2	33.5	0.4	8.6
Oct	595	6.5	495	50	0.6	78.0	1.9	16.8	549	41	0.4	81.8	0.9	25.4
Nov	636	7.0	1078	163	1.8	221.7	2.2	50.2	1011	157	1.7	231.3	2.6	61.3
Dec	534	5.9	286	19	0.2	37.2	0.5	7.7	255	15	0.2	33.1	0.4	8.9
Ave.*	**548**	**6.1**	**801**	**92**	**1.0**	**158.4**	**1.8**	**63.7**	**732**	**87**	**0.9**	**133.0**	**1.5**	**28.6**

Ave.*: the average value.

The value $C(A,B)=1$ means that all solutions in B are weakly dominated by A while $C(A,B)=0$ represents the situation when none of the solutions in B are weakly dominated by A. Table 4 displays the values of $C(A,B)$ for all the data instances. We can see that the falling tide algorithm performs much better as all the $C(A,B)$ values are larger than 0.7.

For future reference, Table 5 gives the summary results of 100 and 200 runs with different weight vectors and L_p metrics. Column "Vio" lists the total number of goal violations existing in the initial solution, and column "%" lists the percentage of violations with regard to the total number of soft constraints. Column "$|D|$" lists the size of an obtained Pareto set (i.e. the number of non-dominated solutions it contains). Columns "Min", "Mean" and "Dev" list the statistical results (in terms of the number of violations) for all the non-dominated solutions contained in our approximated Pareto set.

According to the results in Table 5, we can infer that the set of non-dominated solutions generated by our approach should be a good approximation to the true Pareto set. For example, studying the results after 100 runs (i.e. $N_{run}=100$), the percentages of goal violations are on average 1.0% in terms of "Min" and 1.8% in terms of "Mean", which implies that the solution set is a good approximation to the true Pareto set. In addition, the average number of non-dominated solutions in our Pareto set is as many as 801, which means that those solutions are more likely to spread over the whole Pareto solution space. Table 5 also reveals the time-predefined feature of our approach under the multi-objective framework: if we simply run the algorithm for 200 runs, the indices of "Min", "Mean", and "Dev" are all improved further. As a result, the size of our obtained Pareto set (i.e. $|D|$) has reduced from 801 to 732, which means that at least 70 ($=801-732+1$) non-dominated solutions that were previously obtained are now dominated by the newly replenished solutions.

In modern hospital situations, there is normally a general preference ordering over the individual goals. For example, the Dutch hospital under this study regards that the satisfaction of Goals 1–3 is more important than the others. Hence, in real implementation, the hospital administrator may simply use this priority rule as a filter to remove all the solutions that cannot satisfy these three goals. Under this circumstance, the number of solutions in the approximate Pareto set reduces from several hundred to less than one hundred. The number would further reduce to less than 40 if Goal 4 is also included in the filter. The highly-preferable remaining schedules are finally presented to the relevant nurses, who then select their favourite. Obviously, such filtering rules may also be placed in the actual optimization phase to reduce the size of the solution space. However, the rules applied will be different for different hospitals and potentially for different wards so there will not be a single answer.

To have a better understanding of the plotting of our estimated Pareto set, one would naturally want to 'see' this measurement visually. For a bi-objective problem, it is easy to draw a 2-D graph to show the measure. When the number of objectives increases to three, it becomes harder to determine from a 3-D graph whether the Pareto set is a good one. When the number of objectives is larger than three, there is no way to draw such a graph. For our problem, that has as many as nine objectives, one possible solution is to select pairs of conflicting objectives and draw their 2-D graphs. Fig. 4 displays the plots of the non-dominated solutions in terms of objectives 7 and 8, after 200 runs with changing weights and different L_p metrics for all the test instances. The reason we choose these two objectives is that objective 7 can never be fully satisfied for any data instances according to the results of goal programming, and objective 8 is in conflict with objective 7 according to our observations. From Fig. 4, we can see that, for most instances, the Pareto fronts of these two objectives do exist and the non-dominated solutions do spread evenly over the surrounding area of these Pareto fronts. Note that in Fig. 4, some points appear to be dominated by the others, but this does not mean that they are dominated solutions because they do not necessarily dominate each other by any objectives other than 7 and 8.

5. Conclusions

This paper presented a new approach called the falling tide algorithm for the multi-objective optimization of nurse rostering. It employs a goal programming model to produce an ideal

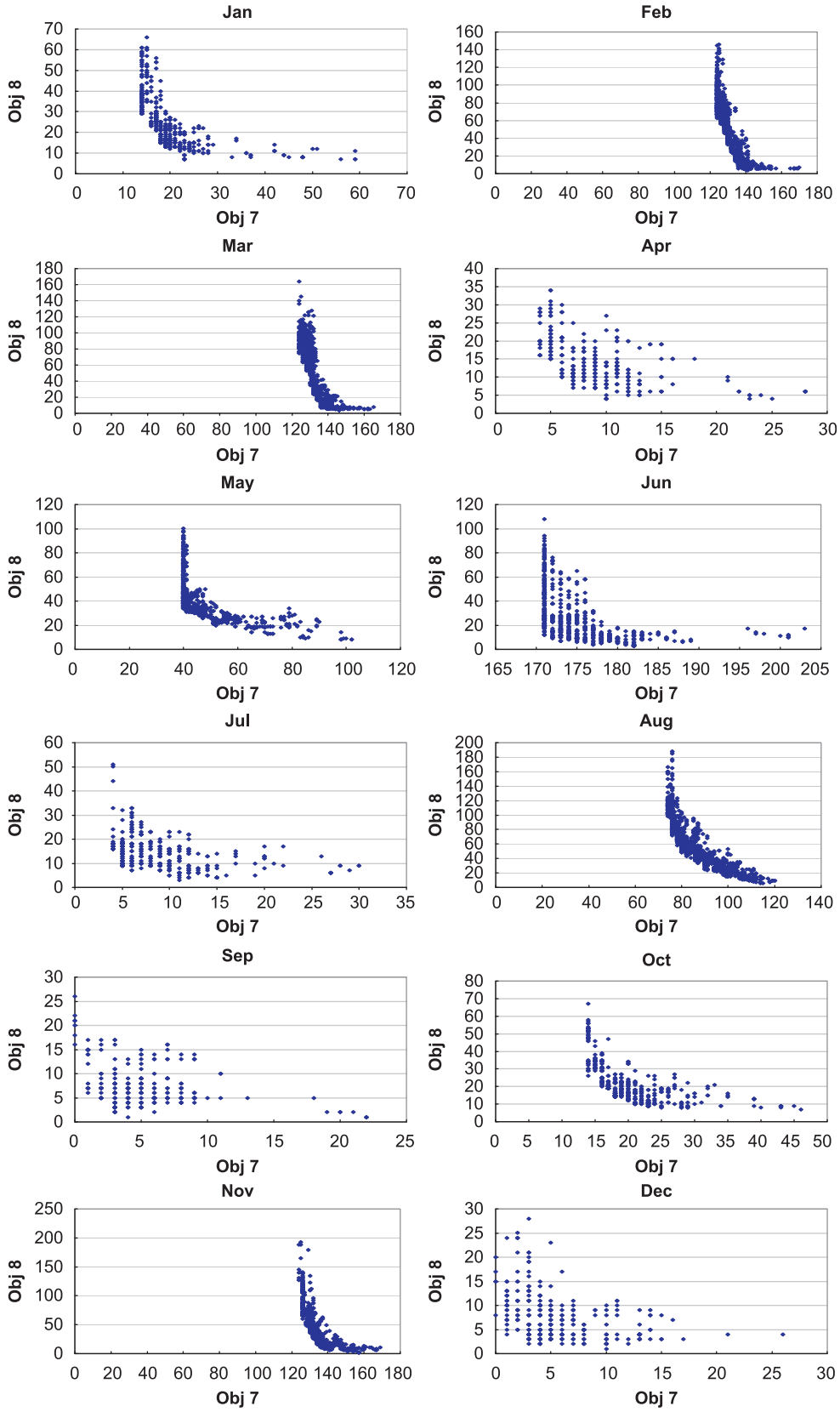

Fig. 4. Plots of the non-dominated solutions in terms of objectives 7 and 8.

objective-value vector and an initial solution as its inputs. The ideal objective-value vector acts as a reference point in a compromise programming based function to evaluate the quality of resulting solutions more efficiently, while the initial solution acts as a good seed for the falling tide algorithm to speed up the convergence.

Experimental results confirm that our approach can give an insight that is not provided by existing approaches. Apart from

another multi-objective approach developed by Burke et al. [15], we do not compare our proposed multi-objective approach to other nurse rostering approaches because the measures used to evaluate the quality of solutions are incomparable. However, we do compare our approach to the others by using the same single weighted cost function, and we achieve significantly better results for 11 out of 12 test instances with less computational time. Hence, we can conclude that our multi-objective approach provides not only high quality solutions, but also enough flexibility in handling different types of constraints that is not possible when using a single objective function.

The major advantage of our approach lies in its ease of implementation and its use of a small number of intuitive parameters that are easy to understand for users. For many types of NP-hard problems, the great deluge algorithm is regarded as a faster and superior variant of simulated annealing. It employs a threshold-based deterministic criterion when accepting/rejecting a non-improved candidate solution compared to the probabilistic one used in simulated annealing. Another important feature of the great deluge algorithm is that it is governed by a single parameter unlike simulated annealing. By inheriting those advantages but overcoming some drawbacks of the great deluge algorithm, we propose a new heuristic search algorithm. This algorithm is easy to understand and flexible to run, and is particularly suitable for various decision support systems where the users prefer to adjust the balance between the execution time and the solution quality in a straightforward manner.

This paper opens up a wide area for further research. In terms of algorithmic improvement, we are looking at other more advanced reductions to achieve more robust performance as our proposed algorithm, in its current form, still uses a linear reduction of the level as the simplest variant.

In terms of potential adaptability, we would evaluate our proposed algorithm in other domains with different types and a different number of constraints. Although the work presented in this paper is on nurse rostering, we anticipate that our proposed approach could be easily adapted to other employee scheduling problems (e.g. audit staff scheduling, call centre rostering, tour scheduling, airline crew scheduling, and transportation driver scheduling) by replacing the current constraints and objective function with the new problem-specific ones. In addition, we anticipate that our approach could be applied to a wider range of other problems (e.g. educational time-tabling problems and resource allocation problems) that are defined by a large number of constraints. When we have many constraints, we often have situations where it is more appropriate to present a user with a range of multi-objective solutions rather than one solution that represents a weighted sum of the objectives. Of course, this is not always the case, but it very often can be.

Acknowledgements

The work was funded by the UK's Engineering and Physical Sciences Research Council (EPSRC), under grant GR/S31150/01.

References

[1] Abraham A, Jain L, Goldberg R. Evolutionary multiobjective optimization.-Springer; 2005.
[2] Aickelin U, Burke EK, Li J. An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. Journal of the Operational Research Society, 10.1007/s10479-009-0590-8, 2009.
[3] Bard J, Purnomo HW. A cyclic preference scheduling of nurses using a Lagrangian-based heuristic. Journal of Scheduling 2007;10:5–23.
[4] Barker TJ, Zabinsky ZB. Multicriteria decision making model for reverse logistics using analytical hierarchy process. Omega—International Journal of Management Science 2011;39:558–73.
[5] Beddoe G, Petrovic S, Li J. A hybrid metaheuristic case-based reasoning system for nurse rostering. Journal of Scheduling 2009;12:99–119.
[6] Beliën J, Demeulemeester EL. Building cyclic master surgery schedules with leveled resulting bed occupancy. European Journal of Operational Research 2006;176:1185–204.
[7] Bowman VJ. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. Lecture notes in economics and mathematical systems, vol. 135, Springer, 1976. p. 76–85.
[8] Brusco MJ, Jacobs LW. A simulated annealing approach to the cyclic staff-scheduling problem. Naval Research Logistics 1993;40:69–84.
[9] Burke EK, Bykov Y, Newall J, Petrovic S. A time-predefined local search approach to exam timetabling problems. IIE Transactions on Operations Engineering 2004;36:509–28.
[10] Burke EK, Cowling P, De Causmaecker P, Berghe GVanden. A memetic approach to the nurse rostering problem. Applied Intelligence 2001;15: 199–214.
[11] Burke EK, De Causmaecker P, Vanden Berghe G. A hybrid tabu search algorithm for the nurse rostering problem. Lecture notes in artificial intelligence, Springer, vol. 1585, 1999. p. 187–94.
[12] Burke EK, De Causmaecker P, Petrovic S, Berghe GVanden. Variable neighborhood search for nurse rostering problems. In: Resende MGC, De Sousa JP, editors. Metaheuristics: computer decision-making (combinatorial optimization book series). Kluwer; 2004. p. 153–72.
[13] Burke EK, De Causmaecker P, Vanden Berghe G, Landeghem H. The state of the art of nurse rostering. Journal of Scheduling 2004;7:441–99.
[14] Burke EK, Kendall G, Soubeiga E. A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 2003;9:451–70.
[15] Burke EK, Li J, Qu R. A Pareto-based search methodology for multi-objective nurse scheduling. Annals of Operations Research 2009, doi:10.1007/s10479-009-0590-8.
[16] Burke EK, Li J, Qu R. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. European Journal of Operational Research 2010;203:484–93.
[17] Chen JG, Yeung T. Hybrid expert system approach to nurse scheduling. Computers in Nursing 1993;11:183–92.
[19] Curtois T. Novel heuristic and metaheuristic approaches to the automated scheduling of healthcare personnel. PhD thesis, School of Computer Science, University of Nottingham, 2007.
[20] Dueck G. New optimization heuristics: the great deluge algorithm and the record-to-record travel. Journal of Computational Physics 1993;104:86–92.
[21] Easton FF, Mansour N. A distributed genetic algorithm for deterministic and stochastic labor scheduling problems. European Journal of Operational Research 1999;118:505–23.
[22] Geldermann J, Bertsch V, Treitz M, French S, Papamichail KN, Hämäläinen RP. Multi-criteria decision support and evaluation of strategies for nuclear remediation management. Omega - International Journal of Management Science 2009;37:238–51.
[23] Geoffrion AM. Proper efficiency and the theory of vector optimization. Journal of Mathematical Analysis and Application 1968;41:491–502.
[24] Jaumard B, Semet F, Vovor T. A generalised linear programming model for nurse scheduling. European Journal of Operational Research 1998;107: 1–18.
[25] Kendall G, Mohamad M. Channel assignment in cellular communication using a great deluge hyper-heuristic. In: Proceedings of the 12th IEEE international conference on networks, 2004. p. 769–73.
[27] Li J, Aickelin U, Burke EK. A component-based heuristic search method with evolutionary eliminations for hospital personnel scheduling. INFORMS Journal on Computing 2009;21:468–79.
[28] Meyer auf'm Hofe H. Solving rostering tasks as constraint optimization. In: Proceedings of the 3rd international conference on practice and theory of automated timetabling. Springer lecture notes in computer science, vol. 2079, 2001. p. 191–212.
[29] Post G, Veltman B. Harmonious personnel scheduling. In: Proceedings of the 5th international conference on practice and automated timetabling, 2004. p. 557–9.
[31] Steuer RE. Multiple criteria optimization: theory, computation and application. New York: John Wiley; 1986.
[32] Zeleny M. Compromise programming. In: Cochrane JL, Zeleny M, editors. Multiple criteria decision making. Columbia: University of South Carolina Press; 1973. p. 262–301.
[33] Zitzler E. Evolutionary algorithms for multi-objective optimization: methods and applications. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.