Combining VNDS with Soft Global Constraints Filtering for Solving NRPs

Jean-Philippe Metivier, Patrice Boizumault, Samir Loudni

Noname manuscript No.
(will be inserted by the editor)

Combining VNDS with Soft Global Constraints Filtering for Solving NRPs

J-P.Métivier · P. Boizumault · S. Loudni

Received: date / Accepted: date

Abstract Nurse Rostering Problems (NRPs) consist of generating rosters where required shifts are assigned to nurses over a scheduling period satisfying a number of constraints. In [25], we have shown how soft global constraints can be used to model NRPs in a concise and elegant way. In this paper we go one step further by proposing new neighborhood heuristics for VNS/LDS+CP. Experiments show that, despite its genericity and flexibility, our approach supplies excellent results on small and middle size problems and very promising results on large scale problems.

Keywords NRP · Constraint Programming · VNS · VNDS · Soft Global Constraints.

1 Introduction

Due to their complexity and importance in real world modern hospitals, Nurse Rostering Problems (NRPs) have been extensively studied in both Operational Research (OR) and Artificial Intelligence (AI) for more than 40 years [5,13]. Most NRPs in real world are NP-hard [20] and are particularly challenging as a large set of different rules and specific nurse preferences need to be satisfied to warrant high quality rosters for nurses in practice. Other wide ranges of heterogeneous and specific constraints usually make the problem over-constrained and hard to solve efficiently [1,32].

NRPs consist of generating rosters where required shifts are assigned to nurses over a scheduling period satisfying a number of constraints [5,10]. These constraints are usually defined by regulations, working practices and preferences of nurses and are usually categorised into two groups: hard constraints and soft constraints (with their violation costs).

Jean-Philippe Métivier
GREYC (CNRS - UMR 6072) – Université de Caen, Campus II – Boulevard du Maréchal Juin, 14000 Caen Cedex, FRANCE
Tel.: + 33 (0)2 31 56 74 84
Fax: + 33 (0)2 31 56 73 30
E-mail: Jean-Philippe.Metivier@info.unicaen.fr

Patrice Boizumaut and Samir Loudni
GREYC (CNRS - UMR 6072) – Université de Caen, Campus II – Boulevard du Maréchal Juin, 14000 Caen Cedex, FRANCE

VNS/LDS+CP [23] is a generic local search method based on VNDS (Variable Neighborhood Decomposition Search [15]) for solving over-constrained problems. Neighborhoods are obtained by unfixing a part of the current solution according to a neighborhood heuristic. Then the exploration of the search space related to unfixed part of the current solution is performed using LDS (Limited Discrepancy Search [16]) combined with Constraint Propagation (Filtering).

Global constraints are often key elements in successfully modelling and solving real-life problems due to their efficient filtering. Global constraints are particularly well suited for modelling (hard) NRP constraints [3,35]. More recently, soft global constraints proposed by [24,31,33,37] enable to quantify the violation while keeping the efficiency of their filtering. In [25], we have shown how soft global constraints can be used to model NRPs in a concise and elegant way. In this paper we go one step further by proposing new neighborhood heuristics for VNS/LDS+CP. Such heuristics provide results better than those described in [25]. Experiments show that, despite its genericity and flexibility, our approach supplies excellent results on small and middle size problems and very promising results on large scale problems.

Section 2 gives a synthetic overview of NRPs. Section 3 describes VNS/LDS+CP and reviews neighborhood heuristics already proposed for solving NRPs using VNS. We performed experiments over different instances selected to be representative of the diversity and the size of NRPs (Section 4). For each selected instance, we compare (Section 5) quality of solutions and computing times for our method with the best known ad'hoc method for solving it [18]. Finally we conclude and draw some further works.

2 Nurse Rostering Problems

2.1 An overview of NRPs

NRPs consist of generating rosters where required shifts are assigned to nurses over a scheduling period (planning horizon) satisfying a number of constraints [5,13]. These constraints are usually defined by regulations, working practices and nurses preference. Constraints are usually categorised into two groups: hard and soft ones.

Hard constraints must be satisfied in order to obtain feasible solutions for use in practice. A common hard constraint is to assign all shifts required to the limited number of nurses.

Soft constraints are not mandatory but are desired to be satisfied as much as possible. The violations of soft constraints in the roster are used to evaluate the quality of solutions. A common soft constraint in NRPs is to generate rosters with a balanced workload so that human resources are used efficiently.

Shift types are hospital duties which usually have a well-defined start and end time. Many nurse rostering problems are concerned with the three traditional shifts Morning, (7:00–15:00), Evening (15:00–23:00), and Night (23:00–7:00). Nurses can possess different skills and cover can be defined for each skill.

Different kinds of constraints can be imposed on the work planning of a nurse:

(i) Shift constraints set the minimal and/or maximal number of nurses of certain skill level working in each shift as well as within each group during the planning period.

(ii) Pattern constraints ensure a certain level of quality for the plannings produced and may be specified either globally for the staff or only for certain individuals. Typical requirements are:

- *patterns of shifts* (i.e. minimal and/or maximal total number of particular sequences of shifts) between any two types of shifts in the planning period (e.g., at least one day-off per week),
- *length of stretches* of shifts of identical type to avoid working too few or too many days in a row on a certain shift (e.g. working more than 4 consecutive day shifts is not permited),
- *patterns of stretches* such as *forward rotation* (going from day shifts to evening shifts to night shifts to day shifts again),
- and *patterns of stretches* of a *given length* that ask for at least so many consecutive shifts of a certain type right after shifts of another type (e.g., there must be a day-off before and after working for three consecutive night shifts).

(iii) Workload constraints are used to model the requirements of min/max total number of hours and/or shifts of specific type worked by each nurse in the planning period.

2.2 Example: `Valouxis` Instance

For the `Valouxis` instance [36], 16 nurses must be planned over a period of 4 weeks. Three shifts are considered: M (Morning), E (Evening) and N (Night). O (Off) will represent repose. The following hard and soft constraints must be enforced. Fig. 1 describes a planning of cost 60 for this instance.

1. **Hard constraints:**
   (H1) From Monday to Friday, M, E and N shifts require respectively (4,4,2) nurses.
   (H2) For weekend, M, E and N shifts require respectively (3,3,2) nurses.
2. **Soft constraints:**
   (S1) For each nurse, the number of M shifts should be within the range [5..8]. Any deviation δ is penalised by a cost $\delta \times 1000$.
   (S2) For each nurse, the number of E shifts should be within the range [5..8]. Any deviation δ is penalised by a cost $\delta \times 1000$.
   (S3) For each nurse, the number of N shifts should be within the range [2..4]. Any deviation δ is penalised by a cost $\delta \times 1000$.
   (S4) Each nurse must have at least 10 days Off. Any shortage δ generates a cost $\delta \times 1000$.
   (S5) Each nurse must have at most 13 days Off. Any excess δ generates a cost $\delta \times 100$.
   (S6) Over a period of 4 weeks, each nurse must have at least 1 free Sunday. Any violation of this rule is penalised by a cost 1000.
   (S7) Each nurse should not work more than 3 consecutive N shifts. Any excess δ generates a cost $\delta \times 1000$.
   (S8) Shift changes must be performed respecting the order: M, E, N. Any violation of this rule is penalised by a cost 1000.
   (S9) Each isolated working day is penalised by a cost 1000.
   (S10) Each isolated day off is penalised by a cost 1000.
   (S11) Each nurse should work 4 consecutive days. Any excess δ generates a cost $\delta \times 1000$. Each period of 2 (resp. 3) consecutive working days is respectively penalized by a cost 40 (resp. 20).

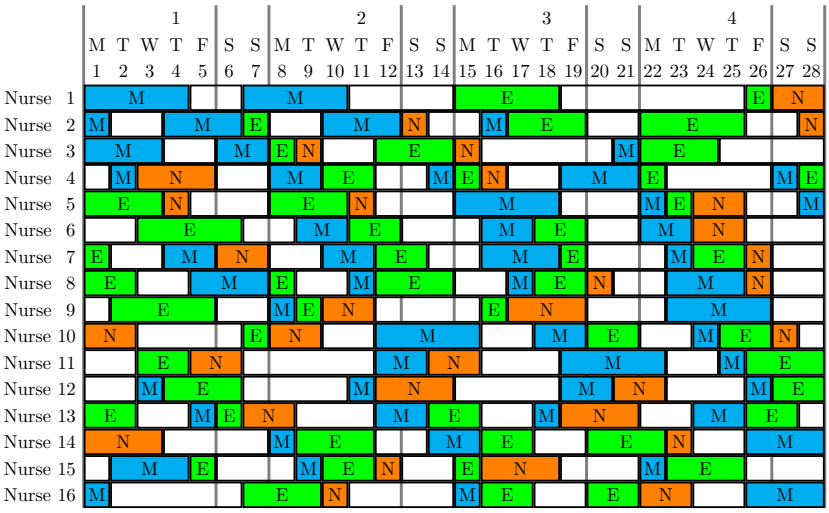

Fig. 1 A solution of cost 60 for the `Valouxis` instance.

3 Solving NRPs

3.1 VNS/LDS+CP

A wide range of approaches and techniques have been proposed for solving NRPs. These include ad'hoc OR methods (by means of mathematical programming with preprocessing steps to reduce the problem size), constructive heuristics and local search methods combining OR techniques to find an initial solution (see [5,13] for a comprehensive review). Of those techniques that have been applied to NRPs, metaheuristics dealing with large-scale neighborhoods (2-opt, swap and interchange of large portions of nurse plannings, ...) such as Variable Neighbourhood Search (VNS) seem to be well suited and very effective.

VNS is a metaheuristic which systematically exploits the idea of large neighborhood change, both in descent to local minima and in escape from the valleys which contain them [27]. Variable Neighborhood Decomposition Search (VNDS) [15] extends basic VNS within a successive approximations method. For a solution of size n, all but k variables are fixed, and VNDS solves a sub-problem in the search space defined by the k unfixed variables.

3.1.1 General overview

VNS/LDS+CP [23] is a generic local search method based on VNDS, where neighborhoods are obtained by unfixing a part of the current solution according to a neighborhood heuristic. Then the exploration of the search space related to the unfixed part of the current solution is performed using a Limited Discrepancy Search (LDS [16]) combined with Filtering in order to benefit from the efficiency of soft global constraints filtering (See Algorithm 1).

Algorithm 1: Pseudo-code for VNS/LDS+CP.

```
   function VNS/LDS+CP(X, C, k_init, k_max, δ_max)
 1 begin
 2     s ← genInitialSol(X)
 3     k ← k_init
 4     while (k < k_max) ∧ (not timeout) do
 5         X_unaffected ← H_neighbor(N_k, s)
 6         A ← s\{(x_i = a) s.t. x_i ∈ X_unaffected}
 7         s' ← NaryLDS(A, X_unaffected, δ_max, V(s), s)
 8         if V(s') < V(s) then
 9             s ← s'
10             k ← k_init
11         else k ← k + 1
12     return s
```

Unlike an usual VNS scheme, our approach offers two main advantages: first, by focusing efforts on improving only a part of the solution, we restrict the size of the search space and intensify search to improve the current solution; second, even if the exploration of (very) large neighborhoods requires a too expensive effort, the use of LDS allows to efficiently explore parts of the search space.

Algorithm 1 shows the general pseudo-code of VNS/LDS+CP, with k_{init} (resp. k_{max}) the minimal (resp. maximal) number of variables to be unassigned and δ_{max} the maximal number of discrepancies allowed for LDS. A subset of k variables (k is the *dimension* of the neighborhood) is selected by the neighborhood heuristic $\texttt{H}_{neighbor}$ in $\mathcal{N}_k$ (set of all subsets of k variables among $\mathcal{X}$) (line 5). A partial assignment $\mathcal{A}$ is generated from the current solution s by unassigning the k selected variables ; the $(n-k)$ non-selected variables keep their current value in s (line 6). Then, unassigned variables are rebuilt by a partial tree search LDS, combined with constraint propagation based on filtering of global constraints. If a solution of better quality s' is found in the neighborhood of s (line 8), then s' becomes the current solution and k is reset to k_{init} (lines 9-10). Otherwise, we look for improvements in the subspace where $(k+1)$ variables will be unassigned (line 11). The algorithm stops when it reaches the maximal dimension size allowed or the timeout (line 4).

3.1.2 LDS+CP

LDS is a tree search method introduced by Harvey and Ginsberg [16] allowing to iteratively solve binary CSPs. Let H be a heuristic that is trusted. The main idea of LDS is to follow H when exploring the search tree, and to consider that H may make mistakes a small number (δ) of times. Thus, δ *discrepancies* are allowed during search. For a given maximal number δ_{max} of discrepancies, LDS explores the tree in an iterative way with an increasing number of *discrepancies* (from $\delta = 0$ to $\delta = \delta_{max}$). Depending on the value of δ_{max}, LDS is either a partial or a complete tree search. In [23], LDS has been extended to n-ary optimization problems, and only performs the last iteration (for $\delta = \delta_{max}$).

Our *variable ordering* for LDS first selects the variable having the lowest ratio domain cardinality divided by its degree (`Dom/Deg`). Our *value ordering* (`BestFirst`) selects the values according to the increasing order of their violation costs. We re-use information gained from the filtering of soft global constraints to determine the

violation cost for a value. Finally, Constraint Propagation is performed using soft global constraints filtering (see [17, 25]).

3.2 Neighborhood Heuristics: Related works

Neighborhood heuristics are crucial since they select parts of the search space to explore in order to find solutions of better quality. However, designing efficient neighborhood heuristics is a hard task and requires a great deal of expertise. Morever, as quoted in [8], few neighborhood heuristics have been designed for NRPs. In this subsection, we review these neighborhood heuristics and describe the context in which they have been used.

(i) (VNS). In [6], three neighborhood heuristics based on swapping large parts of nurse plannings have been proposed and used in a VNS scheme:

- *Shuffle neighborhood* considers different swaps between the worst nurse planning and any other nurse planning.
- *Greedy Shuffle neighborhood* considers swaps between any two nurse plannings.
- *Core Shuffle neighborhood* considers two consecutive swaps between any two nurse plannings at a time (see [6] for more details).

(ii) (VNS+HO). A hybrid method combining VNS with a heuristic ordering (HO) has been proposed in [7]. The aim of the heuristic ordering is to sort all the shifts by their estimated difficulty for assigning them or how likely they are to cause high penalties. First, an initial planning is built using the heuristic ordering. Second, in order to improve the initial planning, a VNS is performed, followed by a repair phase. This phase selects the worst individual plannings, unassigns their shifts and reassigns them using the heuristic ordering. This process is repeated until a stopping criterion is reached. Two kinds of neighborhoods heuristics have been proposed:

- *One-shift Swap*: re-assigning a shift to another nurse working on the same day.
- *Two-shift Swap*: swapping a pair of shifts assigned to two nurses working on the same day.

(iii) (VNS+CP). A 2-steps hybrid Constraint Programming approach has been proposed in [32]. First, a constraint satisfaction model is used to generate weekly plannings of high quality satisfying a subset of shift sequence constraints. An iterative forward search is then used to combine them in order to build feasible solutions over the whole scheduling period (4 weeks). Second, VNS combined with the neighborhood heuristics described in (i) is used to quickly improve obtained solutions.

(iv) (VNS+IP). VNS has been used as a postprocessing step in [10] to make refinements on solutions found by an Integer Program (IP). Proposed neighborhood heuristics are based on swapping groups of consecutive shifts and are very close to the *Greedy Shuffle neighborhood* heuristic decribed in (i).

(v) (LNS). More recently, a LNS (Large Neighborhood Search [34]) scheme has been used to tackle NRPs [17]. It proceeds by selecting fragments of nurse plannings to be unassigned and then rebuilding them using F iltering. Such a use of LNS can be considered as an instance of VNDS. Three neighborhood heuristics have been proposed in [17]:

(a) *Sliding window with a fixed length*: Nurse plannings are selected over a sliding window (i.e. covering fixed days of the roster of all nurses) of one week.

(b) *Sliding window with an overlap*: It is a refinement of heuristic (a) by selecting variables involved in the pattern constraints for which a part of variables is on the boundary of the sliding window, whereas another part is outside the sliding window.
(c) *Detecting regions of low quality*: Instead of selecting all nurse plannings like for heuristics (a) and (b), only nurse plannings of low quality are considered. Moreover, PGLNS [30] is used to determine the size of the sliding window and the set of variables to be unassigned according to the information gained by filtering (see [30] and [17] for more details).

3.3 Neighborhood Heuristics: Our proposal

Neighborhood heuristics based on swap cannot be combined with our VNDS approach which requires an unassigning step and a rebuilding step. Moreover, we haven't used heuristics (a), (b) and (c) described Section 3.2 for two main reasons. First, selecting all nurse plannings is only effective for small problems. For large problems, as neighborhoods size can quickly grow, the exploration of (very) large neighborhoods may require a too expensive effort. Second, as a lot of soft global constraints are stated over the whole planning of a nurse, unassigning only the subset of variables that appear in the sliding window will not lead the rebuilding step to find a new solution of better quality. Indeed, the more the variables are linked, the more opportunities for the rebuilding step to minimize violations.

All variables related to a nurse planning will be together unassigned. For our approach, k will represent the number of nurse plannings to be unassigned (and not the number of variables to be unassigned as depicted in general Algorithm 1). We have considered the following three heuristics:

- `rand` randomly selects nurse plannings,
- `maxV` selects nurse plannings having high violation costs,
- `dilution` combines the two previous heuristics. Among the k nurse plannings to be unassigned, half of them are selected using `maxV`, and the other ones will be chosen randomly. The idea is to mix *intensification* phases (by considering nurse plannings with high violation cost) with *diversification* phases (by considering nurse plannings randomly in order to escape from local minima).

4 Experimental protocol

The ASAP site (Automated Scheduling, optimization And Planning) of University of Nottingham (`http://www.cs.nott.ac.uk/~tec/NRP/`) records a large and various set of NRPs instances as well as the methods used to solve them.

We performed experiments over different instances we selected in order to be representative of the diversity and the size of NRPs (see Table 1). For each instance, **we always compare our approach with the best methods** for solving it [18]. As experiments have been run on various machines, we will report, for each instance, the original CPU time and the processor. For all instances, except the first three ones where the processor is too old to be normalised (they are noted in italic Table 1), CPU times will be normalised[1] and denoted `CPUN`.

[1] For a machine κ times slower than ours, reported CPU times will be divided by κ.

Instances	$\|I\|\times\|J\|$	$\|D\|$	Optimum	Ad hoc methods			VNS/LDS+CP	
				Algo.	**Cost**	**Time(s)**	**Cost**	**Time(s)**
`Ozkarahan`	14×7	3	0⋆	[29]	-	-	**0**	1
`Millar`	8×14	3	0⋆	Network	2550	*500*	**0**	1
				TS+B&B	**0**	1		
`Musa`	11×14	2	175⋆	[28]	199	*28*	**175**	39
`LLR`	26×7	4	301⋆	TS+CP	366	16	312	275
`BCV-5.4.1`	4×28	5	48⋆	Hybrid TS	**48**	5	**48**	1
				VDS	**48**	128		
`Valouxis`	16×28	4	20⋆	VDS	60	32450	60	6570
				SS+VDS	100	6000		
`Azaiez`	13×28	3	0⋆	(0,1)-LGP	**0**	150	**0**	233
`GPOST_A`	8×28	3	5⋆	SS+VDS	9	6457	8	474
				MIP [14]	**5**	1285		
`GPOST_B`	8×28	3	3⋆	SS+VDS	5	5932	4	989
				2-Phases	**3**	14		
				MIP [14]	**3**	441		
`ORTEC_01`	16×31	5	270⋆	GA [7]	775	3600	355	6818
					681	86400		
				VNS+HO [7]	706	3085		
					541	37020		
				VNS+IP [10]	460	2571		
				VDS	355	14359		
					280	51420		
				MIP [14]	**270**	120		
`Ikegami 3Shift-DATA1`	25×30	4	2⋆	TS+B&B	6	111060	63	671

Table 1 Best results for Ad hoc methods vs best results for VNS/LDS+CP.

Some methods include a pre-treatment. As CPU times for this step are not given in papers, reported CPU times concern in fact the second step. In our approach, we use LDS, combined with filtering of soft global constraints, to generate the initial solution. So, reported CPU times for our method always **include the computing time for obtaining the initial solution**.

Benchmarks we considered (see Table 1) represent a wide variety of NRPs with non-trivial properties which are derived from real world complex instances. They are significantly different from each other by the number of nurses (ranging from 4 to 26), the number of shift types (ranging from 2 to 5), the duration of the planning period (ranging from 7 to 31 days) and the constraints to be verified: Shift constraints, Pattern constraints and Workload constraints (see Section 2.1). Finally, they may also differ by the number of personal requests and preferences.

Each instance has been solved by VNS/LDS+CP using neighborhood heuristics `rand`, `maxV` and `dilution`. k_{min} has been set to 2 and k_{max} to 66% of the total number of nurses. Timeout has been set according to the size of each instance. For heuristics `rand` and `dilution`, a set of 10 runs per instance has been performed. VNS/LDS+CP has been implemented in C++. Experiments have been performed under Linux on a 2.8 GHz P4 processor, with 1GB RAM.

5 Experimental results

5.1 Comparing with ad hoc methods

5.1.1 `Ozkarahan` *instance [29]*

We find the optimum in less than 1*s. using* `maxV`*.*

5.1.2 `Millar` *instance (2 methods)*

- *B1) Network programming* [26]: All feasible weekly shift patterns of length at most 4 days are generated. Then, an acyclic graph is defined, where nodes are the stretches, while arcs represent feasible transitions between stretches. Costs are associated to the transitions in order to reflect their desirability. The model is solved using CPLEX.
- *B2) TS+B&B* [19]: Nurse constraints are used to produce all feasible shift patterns for the whole scheduling period for each nurse (independently from shift constraints). Best combinations of these shift patterns are found using mathematical programming and Tabu Search.

With B1, a solution of cost 2,550 is found after 500 s. on an IBM RISC6000/340. With B2, a solution of cost 0 is obtained in 1 s. on a 1GHz Intel P3 processor. *We find the optimum in less than* 1 *s. using* `maxV`*.*

5.1.3 `Musa` *instance [28]*

A solution of cost 199 is found in 28 s. on UNIVAC-1100. *We find the optimum (cost 175) in* 39 *s. using* `maxV`

5.1.4 `LLR instance`

A hybrid AI approach (TS+CP), which combines Constraint Propagation and Tabu Search is used in [22]. First, a relaxed problem which only includes hard constraints is solved as a CSP. Second, adjustments with local search and tabu search is then applied to improve the solution. A solution of cost 366 is found after 96 s. on a PC/P-545MHz (`CPUN` 16 s.). *With* `rand`*, we obtain (on average) a solution of cost* 316.1 *after* 600 *s. The best solution (over the 10 runs) has a cost* 312 *(*275 *s.). The first solution (cost* 363*) is obtained in less than* 1 *s.*

5.1.5 `BCV-5.4.1` *instance (2 methods)*

All the results are obtained on a same machine (2.66GHz Intel P4 processor). Hybrid Tabu search [4] is the best of the 2 methods for this instance. The optimum is found in 5 s. (`CPUN` 5 s.). *With* `dilution`*, we obtain the optimum after* 1 *s.*

5.1.6 `Valouxis` *instance*

This instance [36] is described Section 2.2. In [8], Variable Depth Search (VDS) obtains a solution of cost 60 (3 workstretches of length 3) after 23,175 s. on a 2.66GHz Intel Core2 Duo processor (`CPUN` 32,450 s.). VDS works by chaining together single swaps of shifts among nurse plannings. Several heuristics are used to select the swaps to be chained in order to escape from local optima. In [9], VDS has also been used as an improvement method in the Scatter Search (a population based optimisation method). On this instance, (SS+VDS) obtains a solution of cost 100 in 4,000 s. on a 2.83GHz Intel Core2 (`CPUN` 6,000 s.).

We obtain a solution of cost 60 *(*3 *workstretches of length 3) after* 6,570 *s. using* `rand` *(see Figure 1).*

5.1.7 `Azaiez` *instance*

An optimal solution is provided with the (0,1)-Linear Goal Programming method [2] after 600 s. on a PC/P-700MHz (`CPUN` 150 s.). `rand` *(resp.* `maxV`*) finds the optimum in* 233*s. (resp.* 1,050 *s.).*

5.1.8 `GPOST` *(2 instances)*

The first instance, `GPOST_A`, has an optimal solution of cost 5. The optimum has been found in 1,320 s. using MIP (Mixed Integer Programming) on a P4 2.66GHz (`CPUN` 1,285 s.) This approach [14] takes advantage of the structure of the problem in order to derive new pattern rules to allocate particular shifts e.g. Night shifts. Such propagations drastically reduce the size of the search space. On this instance, (SS+VDS) [9] obtains a solution of cost 9 in 4,305 s. (`CPUN` 6,457 s.).

We find a solution of cost 8 *in* 474 *s. using* `dilution`.

The second instance, `GPOST_B`, is a relaxed version of `GPOST_A` where nurse requests have been removed. For this instance, three approaches have been proposed:

- the same MIP approach [14] finds an optimal solution in 420 s. (`CPUN` 441 s.).
- a 2-steps method [18]. First, all feasible plannings are enumerated for each nurse. Then, the final planning is generated using CPLEX. This method obtains an optimal solution in 8 s. on a 2.83GHz Intel Core2 Duo processor (`CPUN` 14 s.) without taking into account the time used in the first step.
- (SS+VDS) [9] obtains a solution of cost 5 in 3,955 s. (`CPUN` 5,932 s.).

We find a solution of cost 4 *in* 989 *s. using* `rand`.

5.1.9 `Ikegami-3shift-DATA1` *instance*

Experiments have been performed on a P3 1GHz. TS+B&B [19] finds a solution of cost 10 after 543 mns (`CPUN` 194 mns) with a timeout of 24h and a solution of cost 6 after 5,783 mns (`CPUN` 1,851 mns) with a timeout of 100h. `maxV` *provides a solution of cost* 63 *(where all unsatisfied constraints are of weight* 1*) after* 671 *s. with a timeout of* 1*h.*

Contrary to other instances, nurse constraints are hard ones and shift constraints are soft ones for `Ikegami`. So our neighborhood heuristics which unassign whole nurse

Instance	Opt.	First Sol.	rand			maxV		dilution			timeout
			best	time	avg.	best	time	best	time	avg.	
Millar	0	4800	0	2	0	0	1	0	1	0	300
BCV-5.4.1	48	69	48	5	48.8	48	202	48	1	48.6	300
LLR	301	363	312	275	316.1	337	385	315	440	321.3	600
Valouxis	20	37240	60	6570	132	160	3780	60	7160	102	7200
GPOST_A	5	7876	8	654	11.4	14	1252	8	474	11	1800
GPOST_B	3	7362	4	989	8.5	1365	44	5	1701	8.1	1800

Table 2 Comparing heuristics `rand`, `maxV` and `dilution` on several instances.

plannings are irrelevant. If the timeout is increased, the solution quality is improved but it is not enough to bring the optimum. As it is more efficient to unassign variables related to soft constraints than hard ones, one may consider that basic heuristics unassigning shift constraints would be efficient. But it is not the case as it is very difficult to obtain a first solution: the number of nurse constraints is greater than the number of shift ones.

5.1.10 First results for ORTEC_01

The ORTEC_01 instance is a benchmark from ORTEC's Harmony software, an international consultancy company in planning, optimization and decision support solutions. This instance is a large and difficult one. Several approaches have been used to solve it:

- The MIP approach [14] finds an optimal solution in 120 s. (CPUN 120 s.).
- (VNS+HO) [7] finds a solution of cost 706 in 1 h. on a P4 2.4GHz (CPUN 3,085 s.). The same method finds a solution of cost 541 in 12 h. (CPUN 617 mns).
- (VNS+IP) [10] finds a solution of cost 460 in 3,000 s. on a P4 2.4GHz (CPUN 2,571 s.).
- VDS finds a solution of cost 355 in 16,755 s. (CPUN 14,359 s.) and a solution of cost 280 in 60,000 s. (CPUN 51,420 s.) on a P4 2.4GHz.

For a timeout set to $7,200$ s., *we find a solution of cost* 355 *in* 6,818 *s. using* `rand`*, and a solution of cost* 375 *in* 4,231 *s. using* `dilution`*.* More experiments have to be performed to confirm and improve these promising results.

5.2 Comparing our neighborhood heuristics

Table 2 compares the results produced by our neighborhood heuristics (i.e. `rand`, `maxV` and `dilution`) on different instances. For each instance, the cost of the best solution found, its computation time and average solutions over 10 runs are reported. The cost of the first solution we obtained is also recorded in the third column. We can draw some remarks:

- On average, `dilution` outperforms both `rand` and `maxV`, except for LLR, where `rand` is the best one. Indeed, as two consecutive days off get a penalty of 5 and as there are two nurses which require one week day off in their planning leading to a higher violation cost (i.e. 30), heuristics `maxV` and `dilution` will almost select these two nurses, while `rand` will enable to escape from such *local optima*.

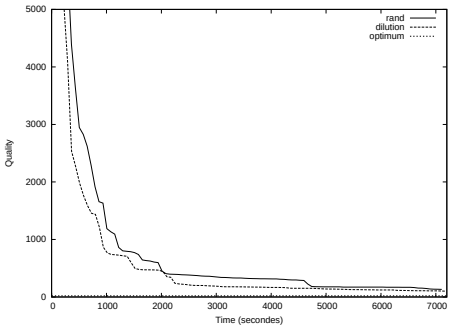

Fig. 2 `Valouxis` Instance, optimum=20

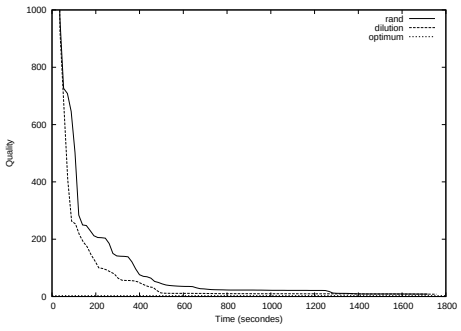

Fig. 3 `GPOST_B` Instance, optimum=3

– For the best reported results, `dilution` and `rand` perform similarly, both in solution quality and computing time. Indeed, when k becomes sufficiently large, the two heuristics tend to have very similar neighborhoods.
– `maxV` is the less effective heuristic. This is probably due to its deterministic criterion, which leads the heuristic to be stucked in local minima. Focussing only on the worst nurse plannings will rarely improve the quality of the overall planning. So, using some randomness enables *diversification*.

The performance profile of a method describes the evolution of the quality of obtained solutions as a function of computation times. Fig. 2 and Fig. 3 depict the performance profiles of VNS/LDS+CP for `Valouxis` and `GPOST_B` instances. As `maxV` is the less effective heuristic, (average) results are only reported for `dilution` and `rand`.

On `Valouxis` instance (see Fig. 2), `dilution` enables to quickly improve the quality of the solution during the search. At the beginning, the performance profile of `dilution` is very close to that of `rand`. But after a few seconds of computation (60 s.) `dilution` always provides solutions of better quality, thus clearly outperforming `rand`. This behavior can be explained by the fact that `dilution` benefits from information provided by `MaxV` tto improve nurse plannings having a high violation cost, but without selecting them all the time.

On `GPOST_B` instance (see Fig. 3), the same conclusions can be drawn: first, solution quality improvements are larger at the early stages of computation (property of *diminishing returns*), particularly during the time interval of [0 . . . 1,000 s.]. Second, the two curves show a decelerating phase leading to a quasi-plateau.

6 Conclusion

For each instance, we have compared our method with the best ad hoc method for solving it [18]. Despite its genericity and flexibility, our method has obtained:

– solutions of better quality and better computing times for `Ozkarahan`, `Millar`, `Musa`, `LLR`, `BCV-5.4.1`, and `Valouxis` ;
– solutions of equal quality with computing times close to those for `BCV541` and `Azaiez`,
– *very promising solution quality* on large scale instances as `GPOST_A`, `GPOST_B` or `ORTEC_01`.

For large instances as `ORTEC` or `Montreal`, or very specific ones as `Ikegami`, performances of our method could be greatly improved by i) using neighborhood heuristics especially designed for NRPs, and ii) reducing the lack of communication between soft global constraints by extending arc consistency for soft binary constraints [11, 12, 21]. In 2009, [21] has shown for the first time that dedicated cost function filtering techniques can also be used to define Global Cost Functions leading to important speedups compared to the use of global constraints with cost variables.

References

1. H. Meyer auf'm Hofe. Solving rostering tasks as constraint optimisation. In *PATAT'00*, LNCS 2079, pages 191–212, 2001.
2. M. Azaiez and S. Al Sharif. A 0-1 goal programming model for nurse scheduling. *Computers and Operations Research*, 32(3):491–507, 2005.
3. S. Bourdais, P. Galinier, and G. Pesant. HIBISCUS: A constraint programming application to staff scheduling in health care. In *CP'03*, volume 2833 of *LNCS*, pages 153–167, 2003.
4. E. Burke, P. De Causmaecker, and G. Vanden Berghe. A hybrid tabu search algorithm for the nurse rostering problem. In *2nd Asia-Pacific Conference on Simulated Evolution and Learning*, volume 1585 of *LNCS*, pages 187–194, 1999.
5. E. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004.
6. E. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. Variable neighborhood search for nurse rostering problems. In *Metaheuristics: computer decision-making*, pages 153–172, Norwell, MA, USA, 2004. Kluwer Academic Publishers.
7. E. Burke, T. Curtois, G. Post, R. Qu, and B. Veltman. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, 188(2):330–341, 2008.
8. E. Burke, T. Curtois, R. Qu, and G. Vanden Berghe. A time predefined variable depth search for nurse rostering, TR-2007-6, University of Nottingham, 2007.
9. E. K. Burke, T. Curtois, R. Qu, and G. Vanden Berghe. A scatter search methodology for the nurse rosetering problem. *Journal of the Operationnal Research Society*, pages 1–13, 2009.
10. E. K. Burke, Ji. Li, and R. Qu. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research*, 203(2):484–493, 2010.
11. M. Cooper, S. de Givry, M. Sanchez, T. Schiex, and M. Zytnicki. Virtual arc consistency for Weighted CSP. In *AAAI'08*, 2008.
12. M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154(1-2):199–227, 2004.
13. A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *EJOR*, 153(1):3–27, 2004.
14. C. A. Glass and R. A. Knight. The nurse rostering problem: A critical appraisal of the problem structure. *EJOR*, 202(2):379–389, 2010.
15. P. Hansen, N. Mladenovic, and D. Perez-Britos. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335–350, 2001.
16. W. Harvey and M. Ginsberg. Limited Discrepancy Search. In *IJCAI'95*, pages 607–614, 1995.
17. F. He and R. Qu. Constraint-directed local search to nurse rostering problems. In *6th International Workshop on Local Search Techniques in Constraint Satisfaction (LSCS 2009), CP'09*, pages 1–12, 2009.
18. *http://www.cs.nott.ac.uk/˜tec/NRP/*.
19. A. Ikegami and A. Niwa. A subproblem-centric model and approach to the nurse scheduling problem. *Mathematical Programming*, 97(3):517–541, 2003.
20. R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.
21. J. Lee and K. Leung. Towards efficient consistency enforcement for global constraints in weighted constraint satisfaction. In *IJCAI'09*, pages 559–565, 2009.

22. H. Li, A. Lim, and B. Rodrigues. A hybrid AI approach for nurse rostering problem. In *SAC*, pages 730–735, 2003.
23. S. Loudni and P. Boizumault. Combining VNS with constraint programming for solving anytime optimization problems. *EJOR*, 191(3):705–735, 2008.
24. J.-P. Métivier, P. Boizumault, and S. Loudni. Softening Gcc and Regular with preferences. In *24th annual ACM Symposium on Applied Computing*, pages 1392–1396, University of Hawaii at Manoa, USA, March 2009.
25. J-P. Métivier, P. Boizumault, and S. Loudni. Solving nurse rostering problems using soft global constraints. In *15th Int. Conf. on Principles and Practice of Constraint Programming*, volume 5732 of *LNCS*, pages 73–87, Lisbon, Portugal, 2009.
26. H. Millar and M. Kiragu. Cyclic and non-cyclic sheduling of 12h shift nurses by network programming. *EJOR*, 104(1):582–592, 1996.
27. N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & OR*, 24(11):1097–1100, 1997.
28. A. Musa and U. Saxena. Scheduling nurses using goal-programming techniques. In *IIE transactions*, volume 16, pages 216–221, 1984.
29. I. Ozkarahan. The zero-one goal programming model of a flexible nurse scheduling support system. In *Int. Industrial Engineering Conference*, pages 436–441, 1989.
30. L. Perron, P. Shaw, and V. Furnon. Propagation guided large neighborhood search. In *Proceedings of CP'04, LNCS 3258*, pages 468–481, 2004.
31. T. Petit, J-C. Régin, and C. Bessière. Specific filtering algorithms for over-constrained problems. In *CP'01*, volume 2239 of *LNCS*, pages 451–463, 2001.
32. R. Qu and F. He. A hybrid constraint programming approach for nurse rostering problems. In *28th SGAI International Conference on A.I.*, pages 211–224, 2008.
33. J-C. Régin, T. Petit, C. Bessière, and J-F. Puget. An original constraint based approach for solving over-constrained problems. In *CP'00*, volume 1894 of *LNCS*, pages 543–548, 2000.
34. P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP'98*, volume 1520 of *LNCS*, pages 417–431, 1998.
35. H. Simonis. Models for global constraint applications. *Constraints*, 12(1):63–92, 2007.
36. C. Valouxis and E. Housos. Hybrid optimization techniques for the workshift and rest assignment of nursing personnel. *A.I. in Medicine*, 20(2):155–175, 2000.
37. W. van Hoeve, G. Pesant, and L-M. Rousseau. On global warming: Flow-based soft global constraints. *Journal of Heuristics*, 12(4-5):347–373, 2006.