An analysis of generalised heuristics for vehicle routing and personnel rostering problems

Mustafa Mısır Pieter Smet Greet Vanden Berghe

CODeS, KAHO Sint-Lieven

CODeS, Department of Computer Science, KU Leuven Campus Kortrijk

{mustafa.misir, pieter.smet, greet.vandenberghe}@kahosl.be

Abstract

The present study investigates the performance of heuristics while solving problems with routing and rostering characteristics. The target problems include scheduling and routing home care, security and maintenance personnel. In analysing the behaviour of the heuristics and determining the requirements for solving the aforementioned problems, the winning hyper-heuristic from the first International Cross-domain Heuristic Search Challenge (CHeSC 2011) is employed. The completely new application of a hyper-heuristic as an analysis tool offers promising perspectives for supporting dedicated heuristic development. The experimental results reveal that different low-level heuristics perform better on different problems and that their performance varies during a search process. The following characteristics affect the performance of the heuristics: the planning horizon, the number of activities and lastly the number of resources. The body of this paper details both these characteristics and also discusses the required features for embedding in an algorithm to solve problems particularly with a vehicle routing component.

Keywords: Hyper-heuristic · Home care scheduling · Security personnel routing and rostering · Maintenance personnel scheduling

Introduction

In the present paper three interrelated problems are investigated:

- The home care scheduling problem (HCSP)
- The routing and rostering of security guards (SPRR)
- The maintenance personnel scheduling problem (MPSP)

These problems all exhibit similar characteristics, the principle being that they are 'structured', i.e. they combine characteristics of different combinatorial optimisation problems. Examples of this kind of problems are the lock scheduling problem (Verstichel & Vanden Berghe, 2009), the tool switching problem (Crama et al., 2007) and the travelling tournament problem (Easton et al., 2001). The structured problem under investigation combine elements from the multi-depot vehicle routing problem with time windows (Cordeau et al., 2002) and personnel rostering (Burke et al., 2004; Ernst et al., 2004). The crux of the problem consists of assigning activities to resources (e.g. visits to patients assigned nurses). Each activity is associated with a geographic location (e.g. the patient's home) and one or more time windows during which the activity should be completed. The goal is to construct efficient routes in which all activities are performed, while also respecting the specified requirements and preferences of the activities where possible. This general integrated problem has been discussed in various domains such as home health care, security and maintenance.

Regarding home health care, the goal is typically to construct shortest routes for nurses in which they visit patients at different geographic locations while respecting a set of constraints (Mısır et al., 2010). In the literature, different versions of this problem have been discussed.

A decision support system in which the utilisation of nurses is maximised over a five day scheduling period is reported on in Begur et al. (1997). Their problem is modelled as a vehicle routing problem with time windows with a central depot and additional constraints concerning route construction, nurse availability and patient visitation requirements. Bertels & Fahle (2006) consider continuity of care as well as staff and patient satisfaction in the evaluation of solutions. The scheduling problems deal with day-to-day operations. Eveborn et al. (2006) describe a successful implementation of a decision support system for scheduling home health care nurses. Besides constraints guaranteeing continuity of care, additional constraints are included in their model to account for various practical restrictions. In contrast to the previously discussed literature, Akjiratikarl et al. (2007) assume that each nurse starts and ends their tour at home rather than in a central depot. Trautsamwieser & Hirsch (2011) address the problem of securing home health care in natural disaster conditions. Particular to their problem definition is that a nurse can work several shifts per day, as long as the workforce related constraints are respected.

In the case of security guard patrols, the focus is more on the routing aspect of the integrated problem rather than on other characteristics such as personnel availability or time windows of jobs. A tabu search metaheuristic used for finding patrol routes on the road network of an estate is presented in Willemse & Joubert (2012). Their model is based on the Chinese postmen and rural postmen problems. Calvo & Cordone (2003) discuss a problem in which both the number of guards and total distance travelled in performing routine inspections should be minimised. Furthermore, some robustness must be incorporated to account for alarm calls. Concerning the staff, there is only one constraint stating that the number of requests assigned to each employee should be limited.

Routing and scheduling in the context of maintenance is closely related to the problem in the home health care context. The emphasis is placed both on routing and personnel rostering. Kovacs et al. (2012) present a problem in which the skill set of employees plays

an important role in deciding which teams are put together whenever a task cannot be completed by a single technician. The objective is to minimise the costs associated with the distance travelled by the teams and the outsourcing of tasks to contractors. In Krumke et al. (2002), both the offline and online version of a routing and scheduling problem at a large car service company are described. Their objective is to minimise the waiting times for customers and keep the operation costs (e.g. tour lengths and overtime expenditure) as low as possible. The considered scheduling horizon also spans one day. Günther & Nissen (2012) focus on determining daily schedules for the technicians in a telecommunication company for delivering service at different locations. The objective function of the problem is used to minimise the travelling time and maximise the completed jobs.

We developed a set of low-level heuristics for solving these problems. The *Generic Intelligent Hyper-heuristic* (GIHH) (Mısır et al., 2012; Mısır, 2012), was used to analyse these heuristics. This algorithm won the first international hyper-heuristic competition (CHeSC 2011). GIHH is a selection hyper-heuristic, which is a high-level search strategy to manage a set of low-level heuristics by using their strengths and eliminating their weaknesses (Burke et al., 2013). The motivation behind hyper-heuristics in this context is to use multiple generalised heuristics and show which heuristics are useful for solving which combined routing and rostering instances. Note that, in the present paper, the focus is not on solving the problem, but rather on providing an analysis of a set of general low-level heuristics for a structured problem by means of a selection hyper-heuristic.

The characteristics of a general routing and rostering problem are presented in Section 2. Alongside this, a discussion of particular properties of the HCSP, SPRR and MPSP is included. Section 3 discusses selection hyper-heuristics in general, the hyper-heuristic used for the experiments and the low-level heuristics. An analysis based on the computational results is presented in Section 4. Furthermore, the results of the analysis are used to construct new low-level heuristics and sets of heuristics which are shown to be more effective than the

complete set of original low-level heuristics. The final section concludes the paper and presents opportunities for further research.

Problem description

This section provides a description of the general problem and further details on the three particular problems discussed in this study. Each of the specific problems introduced in the next sections incorporates characteristics of the general problem while adding additional constraints which differentiate the problems from one another. These additional constraints are what make it hard for a single general purpose approach to find good solutions for each problem.

Followingly, the characteristics of the general problem as well as problem specific constraints are discussed. Additionally, the objective function of each problem is described.

General problem characteristics

Consider the following general, structured problem:

$$\begin{aligned} &\textit{minimise} && f_{\mathrm{VR}}(X) + f_{\mathrm{PS}}(X) \\ &\textit{subject to} && \text{Vehicle routing constraints} \\ & && \text{Personnel scheduling constraints} \end{aligned} \tag{1}$$

Here X represents a solution of the structured problem composed of the vehicle routing and personnel scheduling sub-problems.

The objective to be minimised is composed of two parts, $f_{\mathrm{VR}}(X)$ and $f_{\mathrm{PS}}(X)$, representing the objective functions of the two sub-problems. Note that the combination of the two

objectives is not limited to addition, other operators are also possible. For the general problem, the objective function is a weighted sum of the following: total travel time, idle time and violations of soft constraints. In reference to Equation 1, the first two objectives are part of $f_{\mathrm{VR}}(X)$. The penalty of the soft constraints is contained both in $f_{\mathrm{VR}}(X)$ and $f_{\mathrm{PS}}(X)$. Note that only the components of the objective function are common whereas the weight associated with each soft constraint varies among the different problems and even among instances of a specific problem.

The general model contains two sets of hard constraints, one for each sub-problem. The vehicle routing constraints include common constraints from the vehicle routing literature to guarantee feasible routes, e.g. flow conservation constraints. The personnel scheduling constraints mostly pertain to execution of the activities by the resources, e.g. maximum one activity can be performed by a resource at any one time and an activity locked to a resource cannot be assigned to another resource. Finally, there is a hard constraint that states that all activities must be assigned.

The general problem includes one soft constraint penalising violations of the activities' time windows. This penalty increases linearly with degree to which the time window is violated. For example, if an activity starts 20 time units after the latest start time, a penalty of $20 \cdot w$ is incurred, with w a predefined weight.

The aforementioned constraints form the general problem at hand. Particular problems are usually characterised by additional soft constraints and are discussed for each problem individually in the following sections.

Problem specific extensions

Each particular problem adds additional elements to the general problem discussed in Section 2.1. The problems are distinguished from each other by additional constraints or objectives. Table 1 shows an overview of the different objectives and constraints and whether or not

they are present in each of the problems. In this table, constraints defined as soft constraints are denoted by s, hard constraints are denoted by h.

Home care scheduling

The activities to be performed in the HCSP are patient visits that must be carried out by a set of home care nurses. This problem differs from the other target problems by its emphasis on the patient-nurse relationship, i.e. patients have preferred nurses and vice versa. Also, connected activities, which are activities that should be carried out at the same time or both starting within a defined time interval, occur in the HCSP. Particular care to patients requiring more than one nurse can be modelled using this concept.

There are no additional objectives or hard constraints compared to the general problem. However, there are a number of additional soft constraints. Because of the aforementioned characteristics of the HCSP, violations of nurses' and patients' preferences with regard to one another are considered penalties. Also, the violation of a connected visit incurs a penalty, e.g. when one of two connected visits does not start within the defined time interval. The incurred penalty increases linearly to the amount of violation of the interval. Furthermore, two personnel scheduling soft constraints exist stating that nurses should be qualified for their assigned activities (incurring a fixed penalty each time this is not the case) and that no activities should be assigned to a nurse before or after the start of their shift. The penalty for this latter constraint again increases linearly with the amount of time violated. Finally, there is a soft constraint which penalises whenever priorities assigned to visits are not respected. The incurred penalty for violating this constraint is fixed.

The problem generally addresses day-to-day operations, thus, typically, a planning horizon of only one day is considered. Due to this limited horizon, no complex time-related personnel scheduling constraints should be taken into account. The tested HCSP instances are profiled in Table 2.

Table 1: Overview of problem characteristics. Constraints defined as soft are denoted by s, hard constraints are denoted by h.

	HCSP	**SPRR**	**MPSP**
Objective function			
Min travel time	x	x	x
Min idle/waiting time	x	x	x
Max randomness of routes		x	
Min violations of soft constraints	x	x	x
Vehicle routing constraints			
Assign all tasks	h	h	h
Routes must be feasible	h	h	h
Respect fixed tasks		h	h
Respect task time window	s	s	s
Assign preferred tasks to carers	s		
Respect connected tasks	s		
Respect priorities	s		s
Personnel rostering constraints			
Perform one task at a time	h	h	h
Respect locked tasks	h	h	h
Assign qualified personnel	s	s	h
No overtime	s		s
Assign preferred carers to tasks	s		
Respect max consecutive working days		s	s
Respect max consecutive working time per day		s	s
Respect max working time per week		s	s
Respect max working time per month		s	s
Respect min rest time between two working days		s	s
Respect max consecutive working weekends		s	s
Respect preferred number of working days per week		s	s
No assignments outside feasible regions		s	s

Table 2: The home care scheduling problem instances (*hh*, *ll2*, *ll3*) were taken from Justesen & Rasmussen (2008). The other instances (*hh1*, *ll11*, *ll21*) are the modified versions of the original instances in Justesen & Rasmussen (2008)

Bench.	Number of carers	Number of tasks	Number of patients
hh	15	154	74
hh1	15	150	74
ll11	9	104	59
ll2	7	61	30
ll21	7	60	30
ll3	7	61	30

Routing and rostering of security guards

The routing and rostering of security guards problem (SPRR) concerns the assignment of security tasks to a set of available guards (Mısır et al., 2011a). An intriguing characteristic of this particular problem is that the generated solutions should vary between scheduling periods so that it is difficult to make predictions based on on previous routes. This is a complex feature to include in any model, however, by employing a non-deterministic search algorithm, solutions for different scheduling periods will be structured differently.

One additional hard constraint is added to the general problem model which states that fixed activities should be respected, i.e. jobs a priori assigned a specific start and end time should be respected. Furthermore, a number of personnel scheduling soft constraints are added. These constraints concern assigning qualified personnel to their respective activities (similar to the HCSP), assigning guards to tasks which are on the list of preferred geographic regions and also a set of workforce related constraints, which are shown in detail in Table 3.

Not respecting qualification requirements or preferred regions leads to a fixed penalty, each time a violation occurs. Violations of the workforce related constraints are penalised in proportion to the violation. For example, if a guard works 40 hours in one week while the maximum is 37, the penalty incurred by this violation will be $(40 - 37) \cdot w$ with w the weight of the constraint.

Table 3: Workforce related constraints in the SPRR

Maximum number of consecutive working days	6
Maximum consecutive working time per day (minutes)	720
Maximum working time per week (hours)	37
Maximum working time per month (hours)	175 or 190
Minimum rest time between two working days (hours)	5
Maximum number of consecutive working weekends	2
Preferred number of days worked per week	5

For the SPRR, the scheduling horizon is much longer compared to the other two problems presented and can span up to one month. This is a result of the more predictable nature of the jobs being scheduled. In contrast to patients for home health care, the demand for security jobs typically does not vary much over time, thus allowing for a longer scheduling period to be used.

Table 4 shows characteristics of the SPRR problem instances provided by a software company (Fascinating IT Solutions, `http://www.fit.be`). This dataset is publicly available at `http://allserv.kahosl.be/~pieter/securityguards.html`.

Table 4: The security personnel routing and rostering problem instances

Bench.	Number of guards	Number of tasks
district0	62	1560
district1	348	4217
district2	120	1714
district3	113	2193
district4	192	5252
district5	389	5139

Maintenance personnel scheduling

In the maintenance personnel scheduling problem (MPSP), technicians are to be assigned to various tasks such as installing new devices or repairing malfunctioning infrastructure. Typically these tasks' time windows are defined only by an earliest and latest start date.

Table 5: Workforce related constraints in the MPSP

Maximum consecutive working time per day (minutes)	480
Minimum rest time between two working days (hours)	12
Maximum travelling time between two locations (minutes)	60

This is in contrast to the other problems, where particular times of day are included in the time windows. For example, the time window of a task in the MPSP will state that the task should be performed on either Tuesday or Wednesday. However, in the other two problems, the time windows will also state that a task should be executed between 9:30 a.m. and 11:30 a.m. on either Tuesday or Wednesday. For the MPSP, the decision on the execution time of a task thus mostly depends on the availability of technicians and their workforce related constraints.

This problem adds two additional hard constraints to the general problem. First, the particular date at which a task is to be executed can be determined a priori. If this is the case, the predefined date of execution should be respected. Second, each assigned technician should have the required qualifications. Besides the hard constraints, the MPSP includes two additional soft constraints. The first one states that technicians should only work within their feasible geographic regions. The second presents a set of workforce related restrictions, as shown in Table 5. Violations of the soft constraints are penalised in the same way as the HCSP and SPRR.

Table 6 lists the tested MPSP instances with their details. These instances are based on real world data provided by the same software company that provided the data for the SPRR instances. The dataset can be obtained from `http://allserv.kahosl.be/~pieter/maintenancesched.html`.

The scheduling horizon of this problem is relatively short, typically one week, due to the unpredictable nature of maintenance tasks. Ordinarily, a task constitutes a call from a customer. To satisfy the customer's request as fast as possible, a visit is scheduled within the

Table 6: The maintenance personnel scheduling problem instances

Bench.	Number of technicians	Number of tasks
inst0	12	141
inst1	12	113
inst2	9	110
inst3	10	100
inst4	11	142

next week. The scheduling horizon is therefore limited to the same week. As a consequence of this relatively short scheduling period the number of workforce related constraints is rather small. Note that we consider here the offline version of the MPSP. Considering the problem description it is clear that the MPSP in an online context presents another relevant problem.

Selection hyper-heuristics

Different algorithms are required to find high quality solutions for different problem domains. This statement also holds for the performance difference between algorithms on a set of instances from only one problem domain. Furthermore, it is possible to observe that different algorithms have varying performance results on distinct parts of a search space while solving a problem instance. Therefore, it is required to apply a method for determining which algorithm works better/best in each search region of a search space. This requirement can be achieved using high-level, problem-independent search and optimisation mechanisms, i.e. hyper-heuristics (Burke et al., 2013). These methods are generally categorised under two main types, namely *selection hyper-heuristics* and *generation hyper-heuristics*. A selection hyper-heuristic chooses one or more heuristics and applies them to one or more solutions at each decision step. Contrastingly, a generation hyper-heuristic aims to automatically build these heuristics. This study is concerned with selection hyper-heuristics. A traditional selection hyper-heuristic framework is illustrated in Figure 1, and as mentioned above, this

type of hyper-heuristics chooses heuristics at each decision step and applies them to the solutions of a certain problem. The new solutions found after applying the selected heuristics are examined by a move acceptance criterion to decide whether to accept or reject them based on some problem-independent information. The key advantage in using selection hyper-heuristics is their easy applicability to different problems with their heuristic sets.

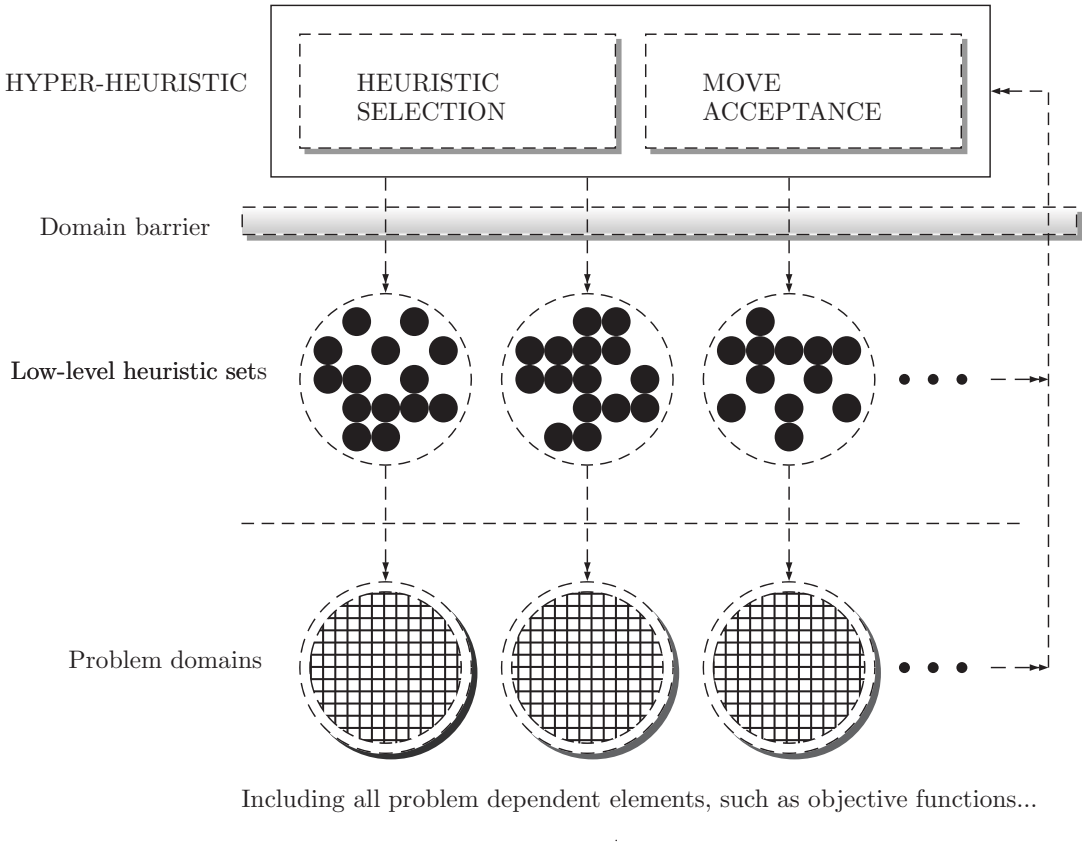

Figure 1: A traditional selection hyper-heuristic

Selection hyper-heuristics for solving various problems have been extensively studied. In Cowling et al. (2001), choice function was introduced to choose heuristics according to their performance and frequency of being called. A study was conducted in Özcan et al. (2010); Nareyek (2003) regarding a suite of reward-penalty strategies in relation to reinforcement learning which utilised certain selection rules in choosing heuristics. Another scoring approach together with a temporary heuristic prohibition mechanism were employed to perform

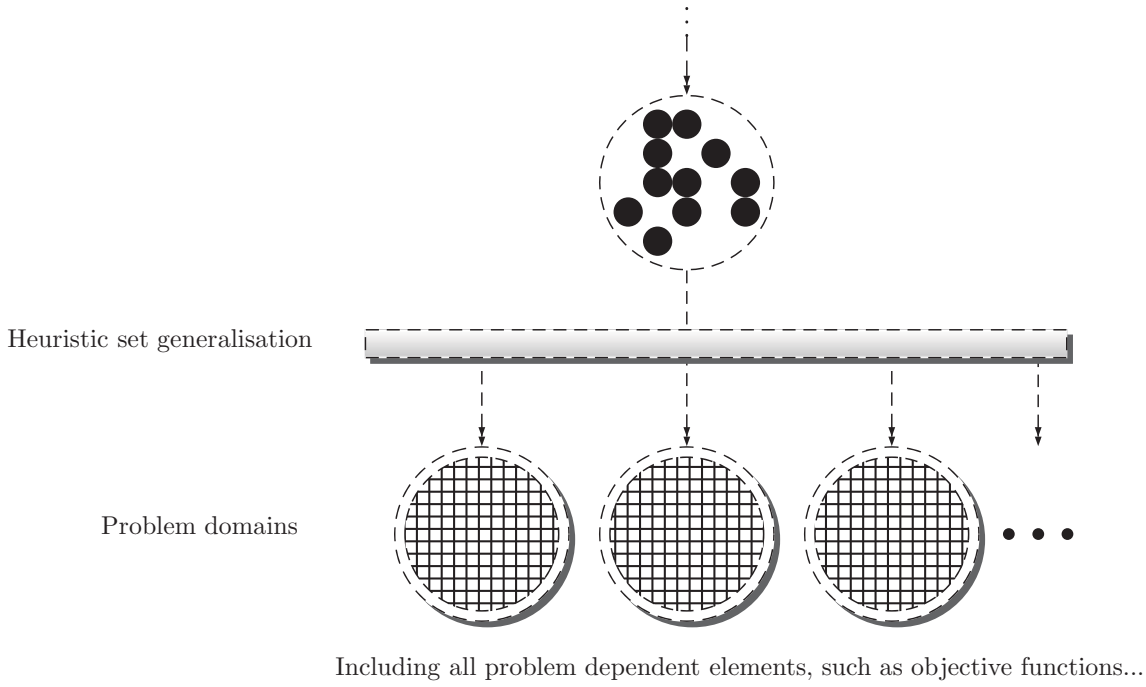

Figure 2: A new selection hyper-heuristic framework

the selection operation in an efficient way in Burke et al. (2003). A reinforcement learning technique, learning automaton, was investigated in deciding which heuristics to apply during the search process (Mısır et al., to appear, 2011b, 2013). Different from these selection approaches, an offline heuristic selection method, case-based reasoning (Burke et al., 2006), chose heuristics using similarities between current and past problem states. Various acceptance mechanisms such as simulated annealing (Bai & Kendall, 2005; Burke et al., 2012), great deluge (Özcan et al., 2010), late acceptance (Özcan et al., 2009), iteration limited list-based threshold accepting (Mısır et al., to appear) were also studied to evaluate the quality of the generated solutions.

In this study, the heuristic set level is also generalised for solving the given routing and rostering problems. In this respect, the heuristic set consists of the operators applicable to any problem from these domains. This provides some kind of a generality for certain problem domains at the heuristic set level, as shown in Figure 2.

An intelligent selection hyper-heuristic as an analysis tool

A selection hyper-heuristic, GIHH (Mısır et al., 2012; Mısır, 2012) performed well across distinct problem domains. GIHH was the winner of the CHeSC 2011 competition by performing as the best solver across 6 problem domains. Therefore, this method was used to analyse the contribution of each heuristic to the hyper-heuristic's performance and to determine the requirements for solving particular instances of the aforementioned problems. GIHH involves a number of sub-mechanisms dedicated to addressing certain problems that can be encountered during the search process. These sub-mechanisms were combined in such a way that they perform in coordination and cooperation thus adapting its behaviour for different environments.

A novel hyper-heuristic sub-mechanism, i.e. ADHS, stands for the adaptive dynamic heuristic set strategy. It aims at specifying heuristic subsets consisting of low-level heuristics which are particularly effective in certain search regions. Resultantly, it adaptively determines heuristic subsets during the run. This process repeat for multiple times during search after (re-)evaluating the performance and behaviour of heuristics. The evaluation considers the improvement capabilities of heuristics together with their speeds. After each evaluation, a score is assigned to each heuristic and the ones with better scores are kept in the heuristic set.

A heuristic is chosen and applied to a solution from these subsets at each decision step. The selection operation is performed based on the selection probabilities of each heuristic, calculated based on pr_i values as shown in Equation 2. $C_{best}(i)$ shows the number of new best solution discovered by heuristic i while t_{spent} shows the time spent while applying this particular heuristic. tf is a time-dependent parameter which linearly decreases from 1 to 0.

$$pr_i = \left((C_{best}(i) + 1)/t_{spent}(i)\right)^{(1+3tf^3)} \tag{2}$$

GIHH additionally provides a relay hybridisation mechanism for identifying proficient heuristic pairs in order to find new best solutions. A fixed-sized heuristic list is attached to each single heuristic to keep track of such pairs. A heuristic list is updated by adding the second heuristic applied after a heuristic if this pair discovered a new best solution. The selection operation among the heuristic pairs is undertaken using a learning automaton (Thathachar & Sastry, 2004). A decision mechanism is employed for choosing between single heuristics and heuristic pairs with respect to the performance of these two heuristic application methods. Algoritm 1 details this process. $C_{best,s}$ and $C_{best,r}$ denote the number of new best solutions delivered by single heuristics and heuristic pairs, respectively. C_{phase} keeps the number of iterations spent within the current phase and pl is the phase length. $(C_{phase}/pl)^{\gamma}$ is used to determine whether to apply heuristic pairs or single heuristics to a solution (S).

Algorithm 1: Relay hybridisation

Input: heuristic list size $list_{size} = 10$; randomly generated values $p, p' \in [0, 1]$

```
1 γ = (C_best,s + 1)/(C_best,r + 1)
2 if p ≤ (C_phase/pl)^γ then
3 |   select LLH using a LA and apply to S → S'
4 |   if size(list_i) > 0 and p' <= 0.25 then
5 |   |   select a LLH from list_i and apply to S' → S''
6 |   else
7 |   |   select a LLH and apply to S' → S''
  |   end
  end
```

Moreover, adaptive iteration limited list-based threshold accepting (AILLA) is used to support these heuristic selection approaches as a move acceptance strategy. Algorithm 2 presents the details of AILLA. AILLA basically keeps track of new best solutions found during earlier iterations and use them as threshold values to accept worsening solutions for the purpose of exploration. The iteration limits, i.e. k and K, are used to determine when to accept a worsening solution and when to change the threshold level. These limits are

adapted during run considering the hardness of the sub-search regions.

Furthermore, a re-initialisation method that speeds the search process up in the case of early convergence to local optima.

Algorithm 2: AILLA move acceptance

Input: $i = 1$, $K \geq k \geq 0$, $l > 0$
for *j=0* **to** *l-1* **do** $best_{list}(j) = f(S_{initial})$
if $adapt\_iterations \geq K$ **then**
  if $i < l - 1$ **then**
    $i++$
  end
end
if $f(S') < f(S)$ **then**
  $S \leftarrow S'$
  $w\_iterations = 0$
  if $f(S') < f(S_b)$ **then**
    $i = 1$
    $S_b \leftarrow S'$
    $w\_iterations = adapt\_iterations = 0$
    $best_{list}.remove(last)$
    $best_{list}.add(0, f(S_b))$
  end
else if $f(S') = f(S)$ **then**
  $S \leftarrow S'$
else
  $w\_iterations++$
  $adapt\_iterations++$
  if $w\_iterations \geq k$ *and* $f(S') \leq best_{list}(i)$ **then**
    $S \leftarrow S'$ and $w\_iterations = 0$
  end
end

The novelty of this paper is applying a selection hyper-heuristic as an analysis tool for a set of generalised low-level heuristics instead of focussing on solving the target problems. This is a more generic contribution that goes beyond the solutions to individual problems. The empirical outcomes can be used to design effective high-level approaches as well as dedicated solvers for the problems on the subject of this study.

Low-level heuristics

12 low-level heuristics, based on move and swap operations, were used for the hyper-heuristic. These heuristics are as follows:

- $\boldsymbol{LLH_0}$: Swap two randomly selected visits between two randomly selected routes.
- $\boldsymbol{LLH_1}$: Swap two randomly selected visits within a randomly selected route.
- $\boldsymbol{LLH_2}$: Swap two groups of consecutive visits from two randomly selected routes.
- $\boldsymbol{LLH_3}$: Move a randomly selected visit in a randomly selected route to another randomly selected route.
- $\boldsymbol{LLH_4}$: Move the most conflicting visit in a randomly selected route to another randomly selected route.
- $\boldsymbol{LLH_5}$: Move the most conflicting visit in a randomly selected route to the least conflicting route.
- $\boldsymbol{LLH_6}$: Move the most conflicting visit in a randomly selected route to a route with the highest idle time.
- $\boldsymbol{LLH_7}$: Move a group of randomly selected consecutive visits to another randomly selected route.
- $\boldsymbol{LLH_8}$: Move a randomly selected visit to another location in a randomly selected route. This heuristic can be considered an Or-opt-1 (Potvin & Rousseau, 1995) move.
- $\boldsymbol{LLH_9}$: Move a group of randomly selected consecutive visits to another location in a randomly selected route. This heuristic can be considered an Or-opt (Potvin & Rousseau, 1995; Bräysy & Gendreau, 2005) move.

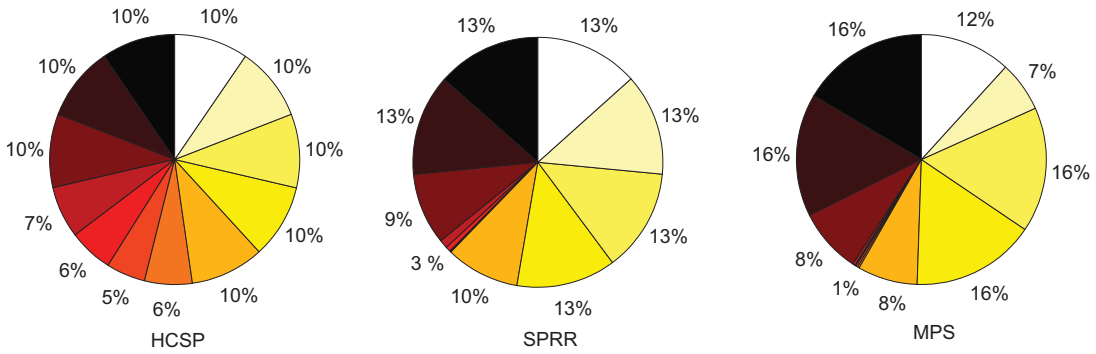

Figure 3: Comparison between the average speed of the low-level heuristics on each problem domain (from black to white in counter-clockwise direction: $LLH_0 \rightarrow LLH_{11}$)

- $\boldsymbol{LLH_{10}}$: Reverse all the visits between two randomly selected visits in a randomly selected route. This heuristic can be considered a 2-opt (Potvin & Rousseau, 1995; Bräysy & Gendreau, 2005) move.

- $\boldsymbol{LLH_{11}}$: Scramble all the visits between two randomly selected visits in a randomly selected route.

These heuristics are responsible for changing the order of visits within a route or between routes. Therefore, after applying each heuristic a function is called to determine the best and earliest possible time assignments for each visit.

Figure 3 presents the speed of these heuristics with respect to each other across the three aforementioned problem domains. For the HCSP, the speed of the heuristics is similar, but the speed of certain move operators is almost half when compared to the rest. The speed difference is greater in the case of the SPRR and MPSP and this occurs mainly due to the required time to perform these moves when working with larger data sets. As shown in Figure 4, the average number of iterations performed by choosing heuristics in a uniformly random manner, also illustrates these differences. The problems, with respect to their speed on executing an optimisation step, can be ordered from fastest to slowest as: HCSP, MPSP, SPRR.

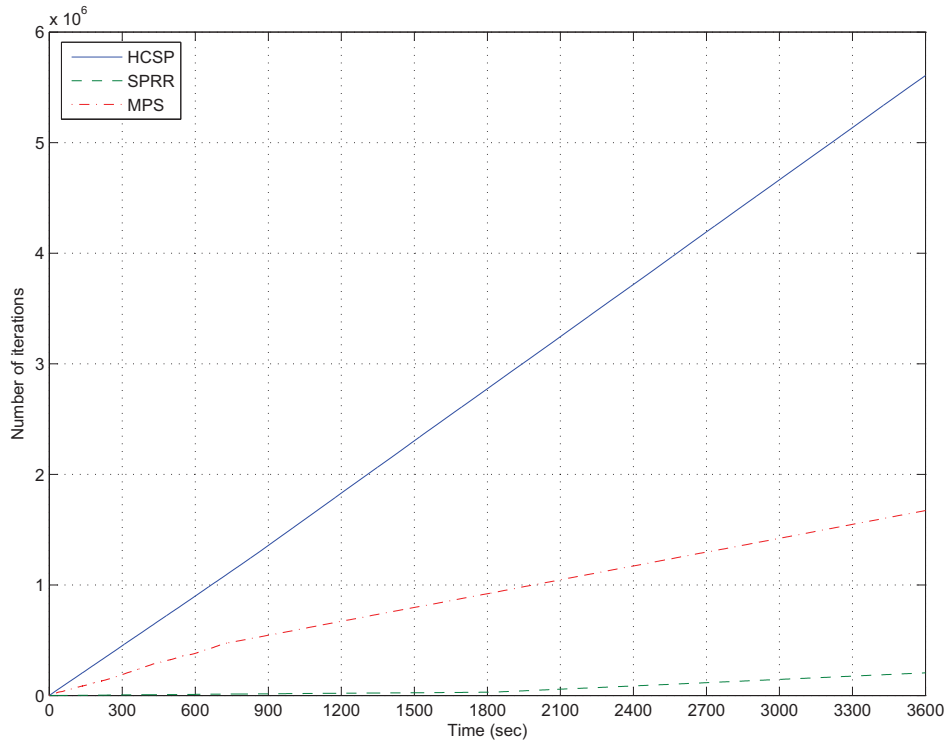

Figure 4: Average operator speed for each problem domain

Computational results

GIHH was applied to each instance 10 times with one hour execution time limits. The experiments were carried out on Pentium Core 2 Duo 3 GHz Pcs with 3.23 GB memory using Windows XP.

Figure 5 provides information on the structure of the search space created by the hyper-heuristic together with the low-level heuristics on each problem. For the HCSP, performing a move can result in major changes on the quality of a solution. This means that, after a high number of improving moves it is possible to visit a solution worse than even a randomly constructed initial solution. Therefore, it is important to develop or use a controlled diversification strategy while solving the HCSP instances. Considering the MPSP, an atomic move in general cannot make a large quality difference with respect to the quality of the visited solutions. During the later iterations, worse solutions can be visited by performing a

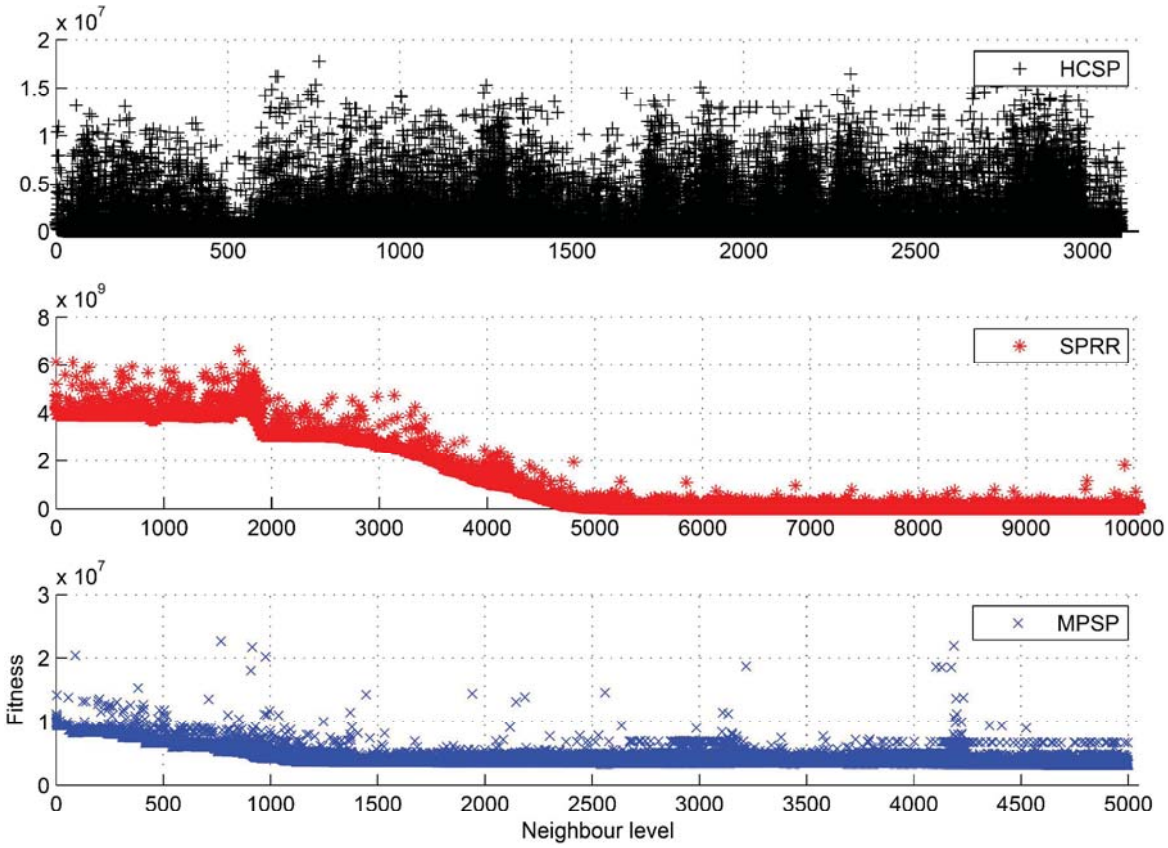

Figure 5: Solutions visited in the course of the search process for each problem (each graph belongs to one run)

move, but this does not occur as often as in the HCSP. The fundamental reason explaining this is the size of the corresponding problem instances. As just mentioned, the HCSP is a daily problem and the number of tasks requiring handling is small compared to the other two problems. The MPSP instances are larger compared to HCSP in terms of the planning horizon, number activities and resources. These elements are even larger for the SPRR case, thus applying a move is less effective with respect to the best and worst solution that can be found. In conclusion, the size of a problem instance is an important element on the performance of a search and optimisation algorithm.

Performance of the low-level heuristics

Each low-level heuristic has certain strengths and weaknesses. It is important to evaluate these features and to perform the heuristic selection process based on them.

Figure 6 illustrates the number of iterations spent by each heuristic throughout the search on the given problems. In the case of HCSP, LLH_3 is the most frequently chosen heuristic by ADHS among those available. LLH_5 and LLH_6 are called less than half of LLH_3. The hyper-heuristic prefers the other heuristics relatively less than these three. For the SPRR, LLH_3 is again preferred for most of the iterations even though LLH_0 is chosen more often at the beginning of the search process. LLH_0 and LLH_4 are also frequently used. The speed of the heuristics has a major effect on the selection process as shown in Figure 3 and 4. The heuristic primarily applied for the MPSP is LLH_1 due to its improvement capabilities. LLH_8 is the other heuristic expected to contribute to the search most.

The number of improving solutions and the number of new best solutions visited by each heuristic are depicted in Figure 7. They show that moving visits from one route to another is the most profitable approach to explore new best solutions of the HCSP and SPRR. However, this is not the case for the MPSP. In this problem, an inner swap move that swaps two visits in a single route delivers a high number of new best solutions when compared with the rest of the heuristics. Considering the personal preferences, contractual constraints and time windows limitations for the HCSP and SPRR, it is possible to understand why move operators between routes work better than the other heuristics. Any violation on such constraints can be resolved or improved by assigning a visit to a proper personel. Related to that, swapping visits internally provide limited improvement. For the MPSP case, swapping can deliver better solutions, since the constraints do not limit the solution space for each technician as much as the HCSP and SPRR do. For instance, time windows for the MPSP are mentioned in the form of days like between Monday and Wednesday while this is in hours-minutes like 08:30-10:00 for the HCSP and SPRR. The possibility of violating time

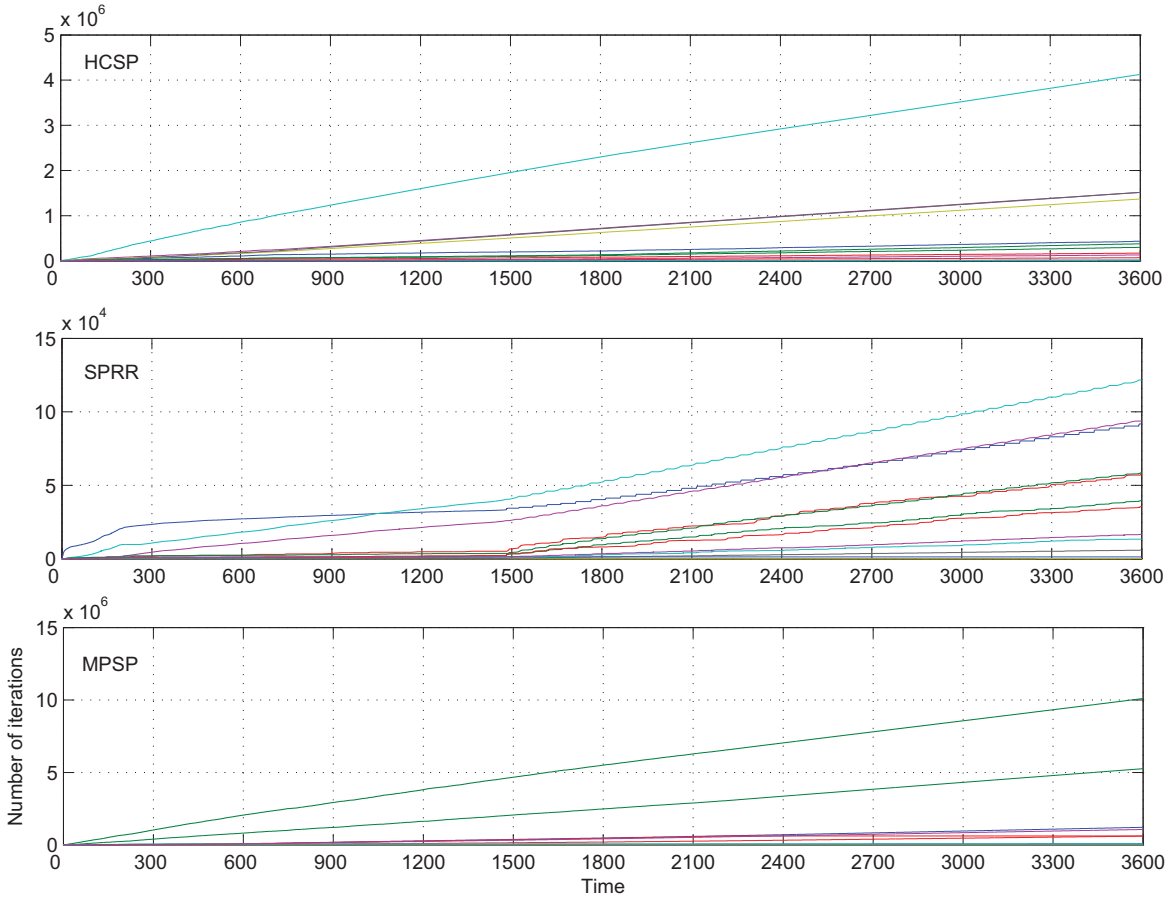

Figure 6: Number of iterations spent per time for each heuristic on each problem domain (each graph belongs to one problem domain ordered as HCSP, SPRR, MPSP from top to bottom)

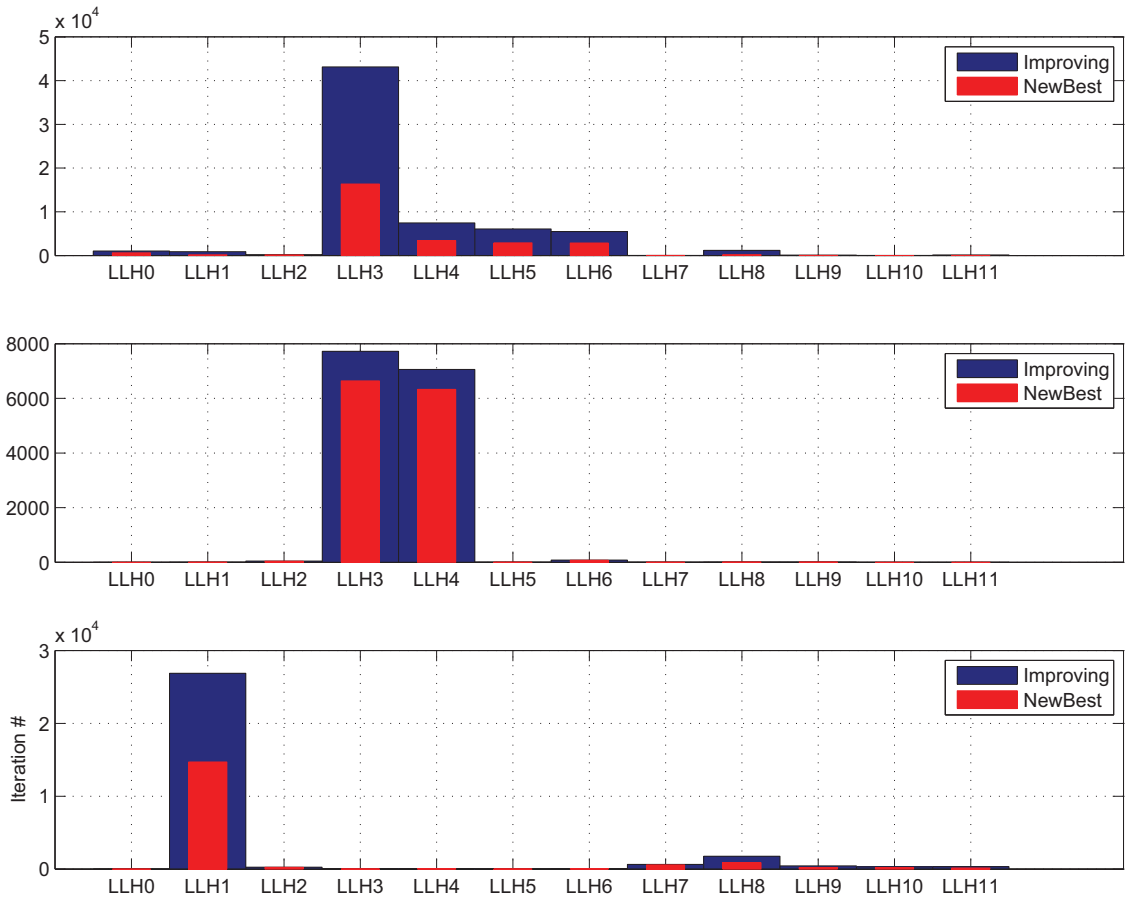

Figure 7: Number of new best and improving solutions found by each heurstic for each problem domain (each graph belongs to one problem domain ordered as HCSP, SPRR, MPSP from top to bottom)

windows due to unavailable technician is less possible compared to the other two problems. This is an important indicator for showing the effect and performance of different heuristics on different problem domains that have common characteristics. Another outcome from this figure is that there are certain heuristics that could not find any new best solution at all. For the HCSP, LLH_{10} exhibits this trait. However, three heuristics, namely LLH_0, LLH_{10} and LLH_{11}, could not succeed in discovering a new best solution for the SPRR. The same situation occurs for LLH_0 while solving the MPSP.

Exploring new heuristics

The performance of a hyper-heuristic depends on a given heuristic set. Hence, its performance is limited to the best performance that can be derived from the heuristic set. Additional strategies can be employed for more efficient usage of the heuristic set instead of only choosing one heuristic and applying it. One way of achieving this aim is finding effective heuristic hybridisations. The employed hyper-heuristic in this study consists of a mechanism to detect good heuristic pairs that can find new best solutions on the run.

Figure 8 depicts the heuristic pairs that found new best solutions for each problem. The squares refer to the first heuristics of each pair while circles denote the second heuristics used. In context of hybridisation, first+second heuristics are considered single heuristics. In other words, when a heuristic pair is called, the first heuristic is applied and the second heuristic is directly applied to the solution found by the first heuristic, in a relay fashion. The majority of these pairs were effective during the first 10 minutes. These heuristic pairs could not deliver new best solutions during the rest of the search process due to the performance of the single heuristics and the evolvability of the search spaces. LLH_6 performed effectively as the secondly applied heuristic in the heuristic pairs for the HCSP. LLH_4, LLH_6 and LLH_8 found solutions that can be easily improved by the second heuristics. The heuristic pairs detected for this problem are $LLH_4 \rightsquigarrow LLH_3$, $LLH_4 \rightsquigarrow LLH_6$, $LLH_6 \rightsquigarrow LLH_3$, $LLH_6 \rightsquigarrow LLH_4$, $LLH_6 \rightsquigarrow LLH_5$, $LLH_6 \rightsquigarrow LLH_6$ and $LLH_8 \rightsquigarrow LLH_6$. All the heuristics responsible for moving visits between routes showed good performance for the heuristic hybridisations in case of the HCSP. Move operators also contributed to the hybridisation process for the SPRR. In this case, LLH_4 is used mostly as the second heuristic of a pair. Effective heuristic pairs discovered by the hyper-heuristic are $LLH_3 \rightsquigarrow LLH_3$, $LLH_3 \rightsquigarrow LLH_4$, $LLH_4 \rightsquigarrow LLH_3$, $LLH_5 \rightsquigarrow LLH_4$ and $LLH_6 \rightsquigarrow LLH_4$. The number of hybridised heuristics found appear to be greater for the MPSP. LLH_1, LLH_2, LLH_7 and LLH_8 showed effective performance as second heuristics of the pairs. Differently to the other two problems, swap heuristics

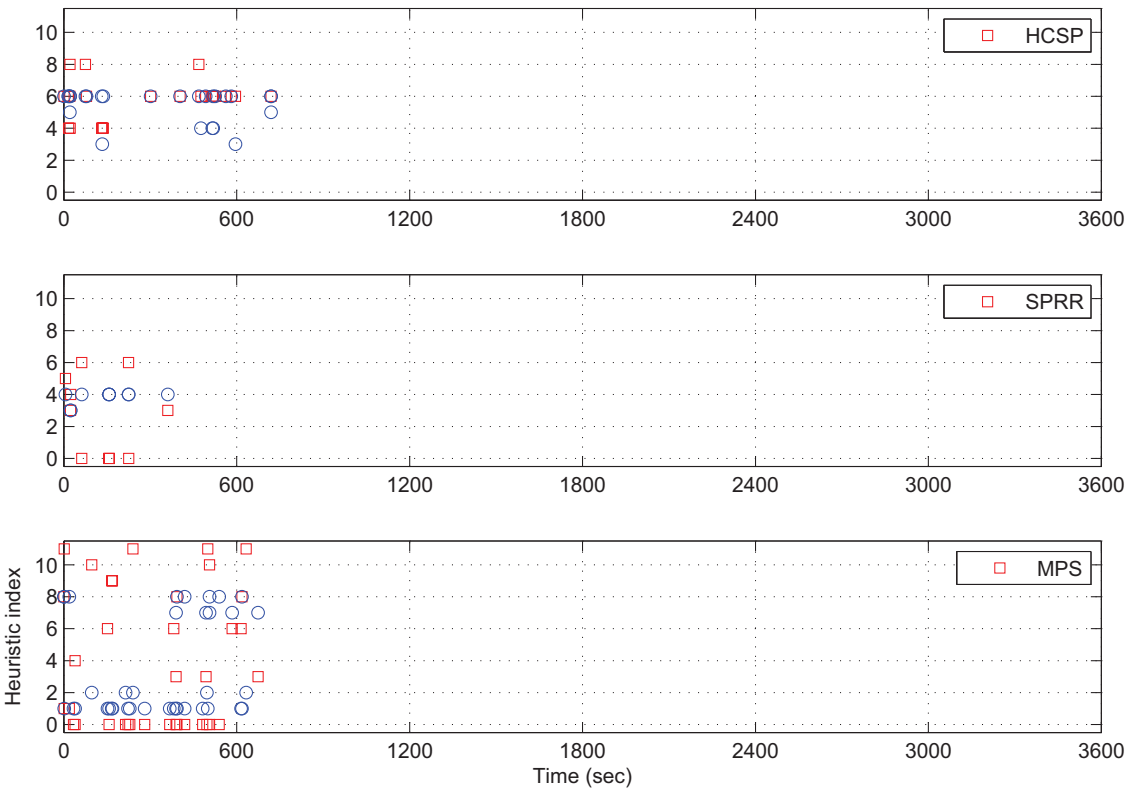

Figure 8: Heuristic pairs found new best solutions (squares and circles represent the consecutively applied heuristics respectively)

performed well in hybridisations. It is possible to see various heuristic combinations with all the heuristics except for when LLH_2 and LLH_5 are the first heuristics for this domain.

These results show that it is possible to find or generate new heuristics based on the given general heuristics for the tested problems. In addition, the heuristics that are not powerful enough alone can be used in combination with other heuristics and can deliver good results. This underlines the importance of using the potential of the different heuristics. More complex hybridisation strategies that can find heuristic combinations of more than two heuristics can be useful for attaining higher performance.

Effect of Offline Learning

The learned information by GIHH is then used to check the performance improvement by the best performing heuristics. A basic and effective hyper-heuristic (SR-IE) involving the Simple Random (SR) heuristic selection mechanism and Improving or Equal (IE) acceptance criterion is used. For the experiments 3 heuristic sets are derived for each problem domain. The first heuristic set, HS_1, consists of all the aforementioned heuristics. The second heuristic set, HS_2, has only the top performing single heuristics. The last heuristic set, HS_3, includes the hybrid heuristics found together with the heuristics from HS_2.

The top performing single heuristics for the HCSP are LLH_3, LLH_4, LLH_5 and LLH_6 as the single heuristics. LLH_3 and LLH_4 are employed as the best single heuristics for the SPRR. The last problem domain, i.e. the MPSP, has LLH_1, LLH_7 and LLH_8 as the top performing single heuristics. All the hybrid heuristics mention in the preceding sub-chapter are used as the hybrid heuristics in the corresponding HS_3 for each problem.

For the HCSP, the best average performance delivered when HS_3 is used on the 5 out of 6 instances. The results on HS_2 show that the top performing heuristics yield relatively worse performance compared to HS_1. This reveals the effectiveness of using the automatically detected hybrid heuristics. For the SPRR, using HS_3 provides better performance on the half of the instances. For the other half, the best results are found when HS_2 is accommodated. The reason behind the superior performance of HS_2 to HS_3 is the limited contribution of the hybrid heuristics as shown in Figure 8. A learning based selection method is expected to deliver better results by choosing single heuristics more frequently after the improvement opportunities by the hybrid heuristic completes.

For the last problem domain, i.e. MPSP, employing the full heuristic set provides superior performance on all the instances. Using HS_3 is better than HS_2 for only one MPSP instance. Despite the worse performance of both HS_2 and HS_3, it should be noted that SR-IE hyper-heuristic fails to manage these heuristics to deliver superior performance than HS_1. This

Table 7: The experimental results by the SR-IE hyper-heuristic after 10 minutes (best average performance for each problem instance shown in colored cells)

	Inst.	HS_1	HS_2	HS_3
HCSP	Inst1	1.64E+06± 2.87E+04	1.64E+06± 2.62E+04	1.45E+06± 8.30E+04
	Inst2	2.45E+06± 2.09E+04	2.61E+06± 1.72E+04	2.15E+06± 2.11E+04
	Inst3	3.58E+06± 5.98E+05	4.44E+06± 1.27E+06	3.59E+06± 9.78E+05
	Inst4	4.02E+05± 6.14E-11	4.03E+05± 0.00E+00	2.11E+05± 3.89E+03
	Inst5	4.02E+05± 6.14E-11	4.03E+05± 0.00E+00	2.09E+05± 3.07E-11
	Inst6	5.28E+05± 1.23E-10	6.74E+05± 1.23E-10	4.24E+05± 0.00E+00
SPRR	Inst1	8.60E+08± 1.34E+08	4.58E+08± 1.19E+08	1.03E+09± 2.00E+08
	Inst2	1.90E+08± 1.07E+08	5.02E+07± 3.41E+07	8.53E+05± 8.81E+05
	Inst3	5.85E+07± 6.97E+07	1.43E+06± 2.72E+06	1.19E+07± 3.34E+07
	Inst4	1.02E+06± 9.23E+05	1.12E+05± 8.54E+04	1.47E+05± 1.04E+05
	Inst5	3.48E+09± 3.35E+08	2.42E+09± 2.25E+08	1.91E+09± 1.16E+08
	Inst6	1.67E+09± 4.96E+08	1.27E+09± 2.33E+08	1.22E+08± 6.28E+07
	Inst7	8.84E+08± 1.12E+08	5.41E+08± 2.31E+07	9.01E+08± 4.41E+07
	Inst8	2.91E+08± 1.29E+08	8.76E+07± 5.35E+07	1.26E+07± 3.16E+07
	Inst9	2.21E+07± 3.41E+07	7.66E+04± 5.13E+04	1.25E+06± 1.33E+06
	Inst10	1.29E+06± 1.63E+06	1.22E+05± 4.02E+04	2.09E+05± 9.58E+04
	Inst11	5.29E+09± 4.35E+08	3.65E+09± 3.01E+08	3.35E+09± 1.04E+08
	Inst12	2.73E+09± 5.77E+08	2.53E+09± 6.90E+08	3.39E+08± 1.25E+08
MPS	Inst1	5.74E+05± 2.07E+05	7.05E+06± 2.96E+06	4.38E+06± 8.37E+05
	Inst2	3.06E+05± 6.00E+04	5.33E+05± 2.27E+05	3.19E+06± 7.50E+05
	Inst3	5.56E+05± 1.66E+05	8.81E+05± 1.72E+05	4.14E+06± 8.23E+05
	Inst4	3.08E+06± 1.52E+05	3.29E+06± 2.27E+05	5.75E+06± 7.64E+05
	Inst5	1.68E+06± 2.39E+05	1.81E+06± 1.84E+05	6.21E+06± 6.26E+05

illustrates the requirement of using a learning based selection method particularly when the heuristic set is relatively larger.

Figure 9 depicts the average ranking when these 3 heuristic sets are used with SR-IE. Across all the problems, HS_3 shows the best (smallest) rank considering all the instances and HS_2 comes second. When each problem is separately considered, the information derived from Table 7 holds. This indicates that if a hyper-heuristic without online learning, i.e. SR-IE, can provide the best performance on average, supporting the derived offline information by online adaptation can yield even better results.

Conclusion

In the present study, three particular problems all based on the same general model were presented, i.e. home care scheduling, security guard routing and rostering and maintenance

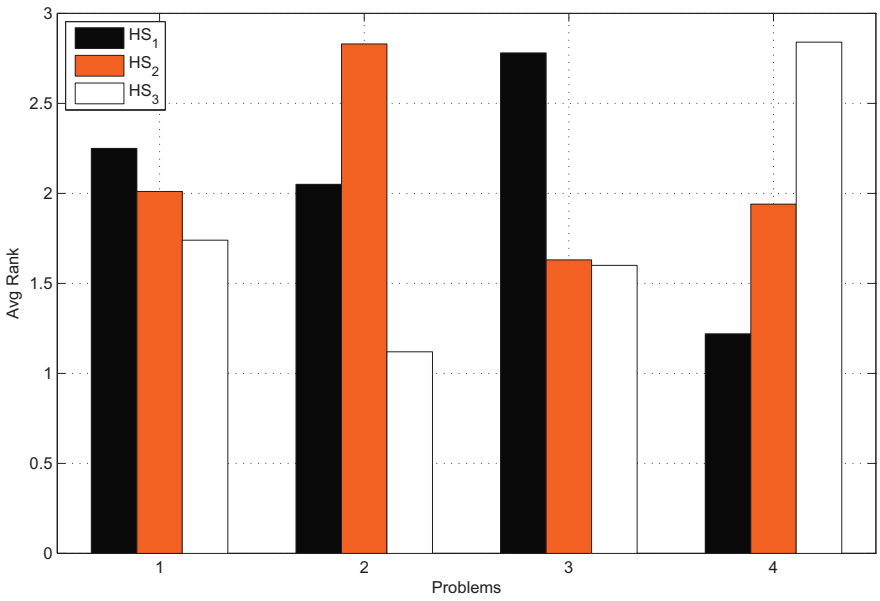

Figure 9: Average ranking of each heuristic set across the corresponding problem instances. In the graph, 1: all instances, 2: HCSP instances, 3: SPRR instances, 4: MPSP instances

personnel scheduling. All the problems incorporate some common objectives as well as the same constraint set. Looking at the characteristics of the three problems it is worthwhile noting that workforce related restrictions are often presented as soft constraints. This conclusion is supported by the large body of literature concerning personnel rostering, in which most workforce related constraints are considered soft. Each of the problems has a particular planning horizon, e.g. one day, one month or one week. This paper defined the set of heuristics that can be used in solving these problems in both a routing and rostering sense. These heuristics involve certain operators that move and swap visits between different routes and make inner route changes. The aim of this study is to show the requirements for solving different problems from routing and rostering domains and to answer the question of how to incorporate multiple generalised heuristics in the context of hyper-heuristics. In answering this question, the hyper-heuristic that won the CHeSC 2011 competition was employed. The experimental results demonstrated that, although the problems have several common features, the heuristics that contribute most to the best resulting solution differ. One major

reason is in connection with certain problem characteristics, namely the planning horizon, the number tasks and the available personnel to handle these tasks. Another outcome of these results is the opportunity of using a selection hyper-heuristic as an analysis tool to study the correlation between algorithms and problems. The proposed methodology can be used to develop new approaches using the gathered knowledge about available heuristics for solving instances from any target problem domain.

Future research plans possible from this paper include: implementing additional generalised heuristics for the target of this study; extending the problem domains using other problems having a certain degree of routing and rostering aspects; developing more intelligent and effective hybridisation mechanisms to generate new heuristics. In addition, this analysis will be repeated for other challenging problem domains.

References

Akjiratikarl, C., Yenradee, P., & Drake, P. (2007). PSO-based algorithm for home care worker scheduling in the UK. *Computers and Industrial Engineering*, *53*(4), 559–583.

Bai, R., & Kendall, G. (2005). An investigation of automated planograms using a simulated annealing based hyper-heuristics. In T. Ibaraki, K. Nonobe, & M. Yagiura (Eds.) *Metaheuristics: Progress as Real Problem Solvers, Selected Papers from the 5th Metaheuristics International Conference (MIC'03)*, (pp. 87–108). Springer.

Begur, S., Miller, D., & Weaver, J. (1997). An integrated spatial DSS for scheduling and routing home-health-care nurses. *Interfaces*, *27*(4), 35–48.

Bertels, S., & Fahle, T. (2006). A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, *33*(10), 2866–2890.

Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Science*, *39*(1), 104–118.

Burke, E., De Causmaecker, P., Vanden Berghe, G., & Van Landeghem, H. (2004). The state of the art of nurse rostering. *Journal of Scheduling*, *7*(6), 441–499.

Burke, E., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., E.Özcan, & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, *64*, 1695–1724.

Burke, E., Kendall, G., Mısır, M., & Özcan, E. (2012). Monte carlo hyper-heuristics for examination timetabling. *Annals of Operations Research*, *196*(1), 73–90.

Burke, E., Kendall, G., & Soubeiga, E. (2003). A tabu-search hyper-heuristic for timetabling and rostering. *Journal of Heuristics*, *9*(3), 451–470.

Burke, E., Petrovic, S., & Qu, R. (2006). Case based heuristic selection for timetabling problems. *Journal of Scheduling*, *9*(2), 115–132.

Calvo, R. W., & Cordone, R. (2003). A heuristic approach to the overnight security service problem. *Computers & Operations Research*, *30*(9), 1269–1287.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M., & Soumis, F. (2002). *The Vehicle Routing Problem*, vol. 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chap. The VRP with Time Windows, (pp. 157–193).

Cowling, P., Kendall, G., & Soubeiga, E. (2001). A hyperheuristic approach to scheduling a sales summit. In E. K. Burke, & W. Erben (Eds.) *Selected Papers from the 3rd International Conference on Practice and Theory of Automated Timetabling (PATAT'00)*, vol. 2079 of *LNCS*, (pp. 176–190). London, UK: Springer-Verlag.

Crama, Y., Moonen, L., Spieksma, F., & Talloen, E. (2007). The tool switching problem revisited. *European Journal of Operational Research*, *182*(2), 952–957.

Easton, K., Nemhauser, G., & Trick, M. (2001). The traveling tournament problem description and benchmarks. In T. Walsh (Ed.) *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming (CP'01)*, vol. 2239 of *LNCS*, (pp. 580–584). Springer.

Ernst, A., Jiang, H., Krishnamoorthy, M., & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, *153*(1), 3–27.

Eveborn, P., Flisberg, P., & Ronnqvist, M. (2006). Laps care–an operational system for staff planning of home care. *European Journal of Operational Research*, *171*(3), 962–976.

Günther, M., & Nissen, V. (2012). Application of particle swarm optimization to the british telecom workforce scheduling problem. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT'12)*, (pp. 242–256). Son, Norway.

Justesen, T., & Rasmussen, M. (2008). *The Home Care Crew Scheduling Problem*. Master's thesis, .U. Denmark & U. of Copenhagen.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, *15*(5), 579–600.

Krumke, S. O., Rambau, J., & Torres, L. M. (2002). Online-dispatching of automobile service units. Tech. Rep. 02-44, ZIB, Takustr.7, 14195 Berlin.

Mısır, M. (2012). *Intelligent Hyper-heuristics: A Tool for Solving Generic Optimisation Problems*. Ph.D. thesis, Department of Computer Science, KU Leuven.

Mısır, M., Smet, P., Verbeeck, K., & Vanden Berghe, G. (2011a). Security personnel routing and rostering: a hyper-heuristic approach. In *Proceedings of the 3rd International Conference on Applied Operational Research (ICAOR'11)*, vol. 3 of *LNMS*, (pp. 193–205). Istanbul, Turkey.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2010). Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'10)*, (pp. 2875–2882). Barcelona, Spain.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2011b). A New Hyper-heuristic Implementation in HyFlex: a Study on Generality. In J. Fowler, G. Kendall, & B. McCollum (Eds.) *Proceedings of the 5th Multidisciplinary International Scheduling Conference: Theory & Applications (MISTA'11)*, (pp. 374–393). Phoenix/Arizona, USA.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2012). An intelligent hyper-heuristic framework for CHeSC 2011. In Y. Hamadi, & M. Schoenauer (Eds.) *Proceedings of the 6th Learning and Intelligent OptimizatioN Conference (LION'12)*, vol. 7219 of *LNCS*. Springer.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2013). A new hyper-heuristic as a general problem solver: an implementation in HyFlex. *Journal of Scheduling*, *16*(3), 291–311.

Mısır, M., Wauters, T., Verbeeck, K., & Vanden Berghe, G. (to appear). A hyper-heuristic with learning automata for the traveling tournament problem. In *Metaheuristics: Intelli-*

gent Decision Making, the 8th Metaheuristics International Conference - Post Conference Volume.

Nareyek, A. (2003). *Metaheuristics: Computer Decision-Making*, chap. Choosing search heuristics by non-stationary reinforcement learning, (pp. 523–544). Kluwer Academic Publishers.

Özcan, E., Bykov, Y., Birben, M., & Burke, E. (2009). Examination timetabling using late acceptance hyper-heuristics. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'09)*, (pp. 997–1004).

Özcan, E., Mısır, M., Ochoa, G., & Burke, E. (2010). A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. *International Journal of Applied Metaheuristic Computing*, *1*(1), 39–59.

Potvin, J.-Y., & Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, *46*(12), 1433–1446.

Thathachar, M., & Sastry, P. (2004). *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. Kluwer Academic Publishers.

Trautsamwieser, A., & Hirsch, P. (2011). Optimization of daily scheduling for home health care services. *Journal of Applied Operational Research*, *3*(3), 124–136.

Verstichel, J., & Vanden Berghe, G. (2009). A late acceptance algorithm for the lock scheduling problem. In S. Voss, J. Pahl, & S. Schwarze (Eds.) *Logistik Management*, (pp. 457–478). Physica-Verlag HD.

Willemse, E. J., & Joubert, J. W. (2012). Applying min-max k postmen problems to the routing of security guards. *Journal of the Operational Research Society*, *63*(2), 245–260.