A Hyper-Heuristic Using GRASP with Path-Relinking: A Case Study of the Nurse Rostering Problem

He Jiang, Dalian University of Technology, China

Junying Qiu, Dalian University of Technology, China

Jifeng Xuan, Dalian University of Technology, China

ABSTRACT

The goal of hyper-heuristics is to design and choose heuristics to solve complex problems. The primary motivation behind the hyper-heuristics is to generalize the solving ability of the heuristics. In this paper, the authors propose a Hyper-heuristic using GRASP with Path-Relinking (HyGrasPr). HyGrasPr generates heuristic sequences to produce solutions within an iterative procedure. The procedure of HyGrasPr consists of three phases, namely the construction phase, the local search phase, and the path-relinking phase. To show the performance of the HyGrasPr, the authors use the nurse rostering problem as a case study. The authors use an existing simulated annealing based hyper-heuristic as a baseline. The experimental results indicate that HyGrasPr can achieve better solutions than SAHH within the same running time and the path-relinking phase is effective for the framework of HyGrasPr.

Keywords: GRASP, Heuristics, Hyper-Heuristics, Nurse Rostering Problem, Path-relinking

INTRODUCTION

Hyper-heuristics aim to design general solving technologies for various problems by choosing existing heuristics (Burke, Hyde, Kendall, Ochoa, Ozcan, & Woodward, 2010). In contrast to meta-heuristics focused on the domain knowledge, hyper-heuristics tend to produce the High Level Heuristics (HLHs) for guiding the Low Level Heuristics (LLHs) (Burke, Hyde, Kendall, Ochoa, Ozcan, & Qu, 2010). The high level heuristics are referred to the heuristics designed by algorithm experts over the problem domains while the LLHs are referred to the heuristics designed by the problem domain experts. Since the domain knowledge is necessary for a particular problem and is hard to explore by an algorithm designer (Ochaoa, Qu, & Burke, 2009), the primary motivation behind the hyper-heuristics is to help the algorithm designers to jump out of the limit from the problem domain and to produce general

DOI: 10.4018/jitr.2011040103

approaches. Based on the ability of general problem solving, hyper-heuristics have been applied to many kinds of problems, especially NP-hard problems, such as the timetabling (Burke, McCollum, Meisels, Petrovic, & Qu, 2007; Qu & Burke, 2009), the cutting stock (Terashima-Martin, Moran-Saavedre, & Ross, 2005), the workforce scheduling (Remde, Cowling, Dahal, & Colledge, 2006; Remde, Dahal, Cowling, & Colledge, 2009) and the p-median (Ren, Jiang, Xuan, & Luo, 2010).

In general, the goal of a hyper-heuristic is to design HLH to find an optimal LLH sequence, which can generate optimal solutions to the problems. As one kind of heuristics, most of hyper-heuristics draw on the experiments from the existing meta-heuristics, e.g., a simulated annealing based hyper-heuristic (Dowsland, Soubeiga, & Burke, 2007) and a genetic algorithm based hyper-heuristic (Ross, Martin-Blazquez, Schulenburg, & Hart, 2003). However, the kinds of hyper-heuristics are much fewer than those of meta-heuristics. The insufficiency of hyper-heuristics has limited the development of hyper-heuristics (Burke, Hyde, Kendall, Ochoa, Ozcan, & Woodward, 2010).

Greedy Randomized Adaptive Search Procedure (GRASP) with path-relinking is one of the effective meta-heuristics for problem solving (Resende & Ribeiro, 2003). There is no hyper-heuristic based on GRASP with path-relinking. As a typical meta-heuristic, GRASP with path-relinking is an iterative procedure to find the optimal solution. GRASP with path-relinking consists of three phases, such as the construction phase, the local search phase, and the path-relinking phase.

Motivated by the success of this algorithm in meta-heuristics, we propose a Hyper-heuristic using GRASP with Path-Relinking (HyGrasPr) in this paper. Our algorithm, HyGrasPr, generates LLH sequences to produce solutions in an iterative procedure. In each iteration, HyGrasPr builds an initial LLH sequence and applies a local search operator to find a relatively good LLH sequence. To avoid to be trapped as a local optimal LLH sequence, the path-relinking strategy is applied to obtain potential good solutions. To show the experimental results of HyGrasPr, we take the nurse rostering problem as a case study. On this problem, an existing simulated annealing based hyper-heuristic (SAHH) is employed as an experiment baseline. The results indicate that HyGrasPr can achieve better solutions than SAHH within the same running time.

The rest of this paper is organized as follows. First, we give the background of our work. We then propose the details of our HyGrasPr and present the experiments results with a case study on the nurse rostering problem.

BACKGROUND

Hyper-Heuristics and Meta-Heuristics

Hyper-heuristic technology is able to handle a wide range of problem domains rather than current meta-heuristic technology concentrated on a particular problem or a narrow class of problems (Burke, Kendall, Newall, Hart, Ross, & Schulenburg, 2003). A solution of a meta-heuristic is a structure abstracted from the problem domain; on the other hand, a solution of a hyper-heuristic is a sequence of LLHs. Many hyper-heuristics are based on the mechanism from meta-heuristics, e.g., hyper-heuristics based on tabu search for timetabling and rostering (Burke, Kendall, & Soubeiga, 2003), simulated annealing for determining shipper sizes (Dowsland et al., 2007), for automated planograms (Bai & Kendall, 2005), genetic algorithm (bin-packing) (Ross et al., 2003), and for 2D-regular cutting stock problems (Terashima-Martin et al., 2005), genetic programming for two dimensional strip packing (Burke, Hyde, Kendall, Ochoa, Ozcan, & Qu, 2010), and ant colony optimization for project presentation scheduling (Burke, Kendall, Silva, O'Brien, & Soubeiga, 2005) and for p-median by Ren et al. (2010). Moreover, Bai, Burke and Kendall (2008) investigate both the meta-heuristics and hyper-heuristics for fresh produce inventory control and shelf space allocation. Considering the similarity

between meta-heuristics and hyper-heuristics, many experiments from meta-heuristics can be employed to guide the hyper-heuristic design. Thus in this paper, we introduce another meta-heuristic to serve for hyper-heuristics, i.e., GRASP with path-relinking.

GRASP with Path-Relinking

GRASP is a multi-start meta-heuristic, in which each iteration consists a construction phase and a local search phase while path-relinking is a strategy to provide various solutions by "relinking" existing solutions (Resende & Ribeiro, 2003). For some of the applications, path-relinking is combined with GRASP to conduct a post-optimization or an intensification strategy to each local optimum (Festa, Pardalos, Pitsoulis, & Resende, 2006). Path-relinking is an enhancement strategy to build paths between solutions and to find other relevant solutions in the paths. Since time-consuming in practice, path-relinking is not applied at each GRASP iteration, but only periodically (Resende & Ribeiro, 2003).

HYPER-HEURISTIC USING GRASP WITH PATH-RELINKING

Framework

In this section, we present the details of our HyGrasPr. The solution generated by HyGrasPr is an LLH sequence, which can be applied to the problem instance in order to obtain the final solution to the problem. The LLH sequence is generated by an iterative procedure of GRASP with the path-relinking phase. In HyGrasPr, we first build an LLH sequence with the fixed length in GRASP construction phase. Then, we search the neighborhood of the LLH sequence for obtaining the local optimal LLH sequence (Croes, 1958). If an LLH sequence gets the best solution, we store it as the current best LLH sequence. Next, we use the current best LLH sequence and the local optimal LLH sequence to conduct the path-relinking phase. In each iteration, we apply the local optimal LLH sequence to the current solution to find the optimal LLH sequence for the problem. In our framework, we give three parameters, namely the number of iteration, the length of LLH sequence in each iteration, and the controllable parameter for the greedy or random strategies.

The pseudo-code for HyGrasPr is presented in Table 1. The kernel idea of our algorithm is to find the optimal LLH sequence by combining GRASP with path-relinking. In our implementation, we store the current best LLH sequence, which can be used for guiding problem solving in the path-relinking phase. Specifically, after a particular number of GRASP iterations, the local optimal LLH sequence is compared with the current best LLH sequence found by now. If the two LLH sequences are different with each other, we call the path-relinking phase. At the end of each iteration, the current best LLH sequence is updated for the next iteration. Existing work on the path-relinking for the meta-heuristics shows that the input length for the path-relinking can increase the running time (Resende & Ribeiro, 2003). In this work, the length of LLH sequence is determined according to an empirical result.

Each iteration of our HyGrasPr begins with an initial solution S_{begin} as an input and ends with a new solution S_{next} as an output. The new solution S_{next} is obtained by applying an LLH sequence to the initial solution S_{begin}. The goal of the hyper-heuristics is to find an optimal LLH sequence, which is applied to the original problem. We denote the LLH sequence as *Seq*. The final sequence of HyGrasPr is combined by all the LLH sequence obtained for each iteration.

GRASP

Based on an LLH sequence with the length N, the hyper-heuristic may provide various solutions for the original problem. To generate an optimal LLH sequence, the GRASP procedure works with two parameters, namely used to denote the length of the LLH sequence and to conduct the construction of the restricted candidate list.

Table 1. Pseudo-code for HyGrasPr

Procedure GRASP with path-relinking Input: maximum iteration *Iteration*, *Seq* length *N*, parameter used in construction α , call path-relinking frequency *PF* Output: Solution *S*
1 for i = 1,…, *Iteration* do 2 *Seq* = GRASP_Construction(N, α , S_{begin}, S_{next} 3 Seq_{local} = Local_Search(N, S_{begin}, S_{next}, *Seq*) 4 if i % *PF*= 0 then 5 Path-Relinking(Seq_{best}, Seq_{local}, S_{begin}, S_{next}) 6 endif 7 update Seq_{best} 8 $S_{begin} \longleftarrow S_{nextNew}$ 9 endfor

In general, each iteration of GRASP consists of two phases, namely the construction phase and the local search phase. We describe the construction phase and the local search phase for GRASP in our HyGrasPr. The construction phase of HyGrasPr is different from the typical GRASP construction for the metaheuristics. The output of the construction phase is an LLH sequence, which is build from an empty sequence. In each iteration of the construction phase, we add an LLH to the current LLH sequence until the LLH sequence length reaches the particular value *N*. The LLH to be added is randomly selected from the Restricted Candidate List (RCL). The RCL is a subset of the set of all candidate LLHs. If an LLH can generate a solution according with the given threshold, the LLH will be added to the RCL. The threshold is defined as $c^{\min} + \alpha(c^{\max} - c^{\min})$,where $c^{\min}$ and $c^{\max}$ are the minimum and maximum of the evaluation function values for a new solution after applying the LLH to the current solution. The parameter α is a controllable parameter and $0 \le \alpha \le 1$. If $\alpha = 0$, the construction phase can be viewed as a greedy construction while if $\alpha = 1$, the construction phase can be viewed as a random construction. We present the pseudo-code of the construction phase in Table 2.

After the construction phase of the HyGrasPr, we apply a local search phase to optimize the current LLH sequence. The 2-opt search is employed in the local search phase. To find the optimal LLH sequence, each LLH in the *Seq* is traversed to search for the opportunity to build the final optimal LLH sequence. In each step, we change the selected LLH with other candidate LLHs, and then apply the new LLH sequence Seq_{temp} to the solution S_{begin}. If the new Seq_{temp} in the neighborhood could obtain a better solution, we replace Seq_{local} with Seq_{temp}. After applying the current best LLH sequence to the solution S_{begin}, the new obtained solution is used as the initial solution in the next iteration. In Table 3, we present the pseudo-code of the local search.

Path-Relinking

In this section, we propose the path-relinking phase for HyGrasPr. Path-relinking can be viewed as an enhancement to the GRASP procedure. The main idea of the path-relinking is to explore the trajectories that connect an initial LLH sequence to a guiding LLH sequence. The initial LLH sequence is the input of the path-relinking while the guiding LLH sequence is the objective to guide the exploration. In HyGrasPr, a move of the path-relinking phase is

Table 2. Pseudo-code for construction of GRASP in HyGrasPr

Procedure construction Input: LLH sequence length N, parameter α , the initial problem solution S_{begin} Output: LLH sequence *Seq*, a new problem solution S_{next}
1 $Seq \leftarrow \Phi$, 2 best value by LLH sequence of current Iteration $Best \leftarrow \infty$ 3 while the length of $Seq < N$ 4 for every LLH $H_i \in H$(the set of low level heuristics) do 5 apply H_i to current solution $S_{current}$ get new solution S_i 6 C_i = objective value of S_i 7 endfor 8 $c^{\max} \leftarrow \max\{ c_i \mid c_i \in C\}$ 9 $c^{\min} \leftarrow \min\{ c_i \mid c_i \in C\}$ 10 the restricted candidate list $RCL \leftarrow \{i \in H \mid c_i \leq c^{min} + \alpha(c^{max} - c^{min}) \}$ 11 randomly select a heuristic num h from RCL 12 if $c_h < Best$ then 13 $Best = c_h$ 14 endif 15 add h to Seq 16 $S_{current} \leftarrow S_h$ 17 endwhile 18 $S_{next} \leftarrow S_{current}$

defined as a change from one LLH to another for reducing the difference between the guiding LLH sequence and the initial LLH sequence. In our experiments, the initial LLH sequence is the best LLH sequence found up till now and the guiding LLH sequence is the local optimal LLH sequence in current iteration. The initial LLH sequence can be defined as Seq_i and the guiding solution can be defined as Seq_g. We present the pseudo-code for this phase in Table 4.

A CASE STUDY OF THE NURSE ROSTERING PROBLEM

The Nurse Rostering Problem

The nurse rostering problem is a typical problem in the family of the personnel scheduling problem (Burke, Causmaecker, Berghe, & Landeghem, 2004). The goal of the nurse rostering problem is to decide the assignment of each nurse working over a specific planning period subject to various constraints (Burke, Li, & Qu, 2010). In this paper, we take the nurse rostering problem as a case study to investigate the performance of our HyGrasPr. The benchmark instances can be obtained from the nurse rostering problem website (http://www.cs.nott.ac.uk/~tec/NRP/). These instances in our experiments are generated from the real-world scenarios. All the constraints in the nurse rostering problem instances are modeled as the weighted objectives (Curtois, Ochoa, Hyde, & Vazquez-Rodriguez, 2010). The mathematical formulation of these objectives is described in (Burke, Curtois, Qu, & Vanden-Berghe, 2008).

Table 3. Pseudo-code of local search of GRASP in HyGrasPr

Procedure local search Input: LLH sequence length N, LLH sequence Seq, initial problem solution S_{begin}, best value by LLH sequence of current Iteration $Best$, S_{next} got from construction phase Output: local optimal LLH sequence Seq_{local}, problem solution S_{next}^{1}
1 $Seq_{local} \longleftarrow Seq$ 2 for i = 0,...,N do 3 for j = 0,…, H_{num}(the number of low level heuristics) do 4 if the ith LLH of Seq! = j then 5 get a new LLH sequence Seq_{temp} by replacing ith LLH of Seq with j 6 $S_{temp} \longleftarrow S_{begin}$ 7 $localOptimalFlag$ = false 8 for k = 0,…,N do 9 apply kth LLH of S_{temp} to S_{temp} 10 compute the objective value of S_{temp} $value$ 10 if $value$ < $Best$ then 11 $Best$ = $value$ 12 Seq_{local} = Seq_{temp} 13 $localOptimalFlag$ = true 14 endif 15 endfor 16 if $localOptimalFlag$ then 17 $S_{next} \longleftarrow S_{temp}$ 18 endif 19 endif 20 endfor 21 endfor

Table 4. Pseudo-code of path-relinking in HyGrasPr

Procedure path-relinking Input: initial solution Seq_i, guiding solution Seq_g Output: the best LLH sequence Seq_{best}
1 compute the sum $Number$ of different LLHs between Seq_i and Seq_g 2 save the different LLHs in the move set, M 2 for i = 0,…,$Number$ do 3 find the $bestmove$ in the move set, M 4 apply the $bestmove$ 5 update the Seq_{best} 6 update the best problem solution 7 delete the selected $bestmove$ from M. 8 endfor

To show the performance of our HyGrasPr, we use 12 LLHs in this paper. These LLHs are obtained in the hyper-heuristic competition website (http://www.asap.cs.nott.ac.uk/chesc2011/index.html). A detailed description of these heuristic is reproduced here for completeness (Curtois et al., 2010).

- h_1, local search heuristic using "hill climbers" with vertical neighborhood operator (moving shifts vertically between two employees in the roster);
- h_2, local search heuristic using "hill climbers" with horizontal neighborhood operator (moving shifts horizontally in single employee's work pattern in the roster);
- h_3, local search heuristic using "hill climbers" with new neighborhood operator (introducing new shifts or deleting shifts into the roster);
- h_4, local search heuristic, variable depth search using new moves as links in the ejection chain;
- h_5, local search heuristic, variable depth search using new moves as links in the ejection chain and test replacing an entire work pattern for a single employee as a link in the chain;
- h_6, ruin and recreate heuristic, un-assigning $x = Round(\beta \times 4) + 2$ schedules(β is the intensity of mutation and *Round*() is the rounding function). The heuristic first randomly selects x employee's schedules and un-assigning all the shifts in them. They are rebuilt by first satisfying objectives related to requests to work certain days or shifts and then by satisfying objectives related to weekends;
- h_7, same as h6 but "$x = Round(\beta \times$ Number of employees in roster)";
- h_8, same as h6 but creating a small perturbation in the solution by using $x = 1$;
- h_9, crossover heuristic, choosing the best x assignments in each parent and make these assignments in the offspring;
- h_{10}, crossover heuristic, it creates a new roster by using all the assignments made in the parents. It makes those that are common to both parents first and then alternately selects an assignment from each parent and makes it in the offspring unless the cover objective is already satisfied;
- h_{11}, crossover heuristic, creating the new roster by making assignments which are only common to both parents;
- h_{12}, mutation heuristic, randomly mutating a selected roster.

Results

We show the performance of our HyGrasPr in the experiments. All the experiments are run on a PC with Intel Core Duo 2.53 GHz and 4GB RAM running on a Microsoft Windows 7 Ultimate.

Before showing the experimental results, we give some experiments on the parameter values and performance. We test our algorithm on 4 typical problem instances (ORTEC02, Ikegami-3Shift-DATA1, Ikegami-3Shift-DATA1.1, and Ikegami-3Shift-DATA1.2) with different parameters. The parameter α which is the probability of choosing RCL in GRASP is set to 0.3. The parameter N, the length of LLH sequence is set to 5, 10, 15, and 20, respectively. We limit the time bound to 30 minutes. The results are shown in Table 5.

It is showed that when the parameter α is fixed to 0.3, the LLH sequence length with the value 15 gets the best result. In general, for the length from 5 to 15, the objective function of solutions is decreasing when the length increases. That is the length with the value 15 may bring better solutions than others. On the other hand, when the sequence length increases, the running time may increase. Therefore, the experiment with the length of the value 20 can cost more time than the previous three experiments. Since we limit the time bound, the experiment with the length 20 cannot finish the process of searching. Thus, we choose 15 as the sequence length in the following experiments.

We study the parameter α with the values among 0.1, 0.3, 0.5, and 0.7. We also limit the

Table 5. Objective functions for HyGrasPr with different sequence length on 4 typical instances

Sequence length	5	10	15	20
ORTEC02	335.0	350.0	310.0	335.0
Ikegami-3Shift-DATA1	18.0	11.0	17.0	22.0
Ikegami-3Shift-DATA1.1	23.0	21.0	13.0	21.0
Ikegami-3Shift-DATA1.2	20.0	23.0	18.0	20.0

Table 6. Objective functions for HyGrasPr with different α on 4 typical instances

Value of α	0.1	0.3	0.5	0.7
ORTEC02	365.0	310.0	330.0	380.0
Ikegami-3Shift-DATA1	22.0	17.0	13.0	15.0
Ikegami-3Shift-DATA1.1	20.0	13.0	15.0	31.0
Ikegami-3Shift-DATA1.2	19.0	18.0	21.0	23.0

Table 7. Results for SAHH, HyGrasPr, and HyGrasPr without path-relinking

Problem instance	Time(s)	SAHH	HyGrasPr	HyGrasPr without path-relinking
BCV-3.46.1	17870	3321.0	3307.0	3322.0
BCV-A.12.2	12012	2210.0	2005.0	2340.0
ORTEC02	4972	395	300.0	440.0
Ikegami-3Shift-DATA1	5981	12.0	11.0	21.0
Ikegami-3Shift-DATA1.1	6315	15.0	13.0	25.0
Ikegami-3Shift-DATA1.2	5990	15.0	14.0	36.0
CHILD-A2	43331	1103.0	1108.0	1240.0
ERRVH-A	20022	2165.0	2156.0	2344.0
ERRVH-B	16948	3167.0	3167.0	4093.0
MER-A	38369	9289.0	8934.0	14964.0

time bound to 30 minutes. The results are showed in Table 6.

In the structure of the construction phase, the parameter α is a trade-off between greedy and random. From the results, we can see that the value 0.3 maybe a good value for the instances in our experiments.

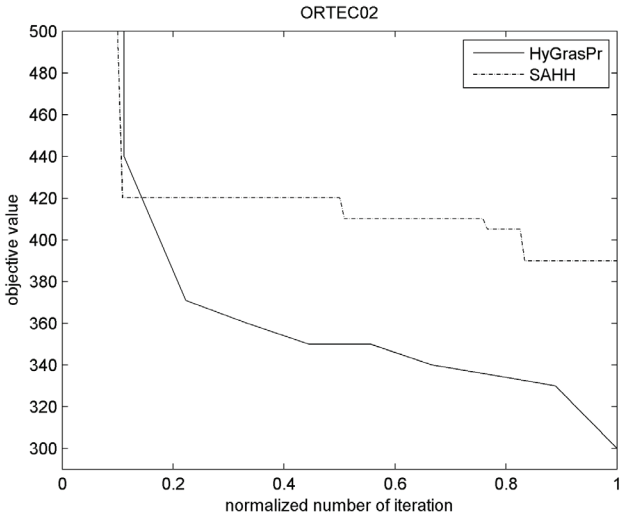
The experiments of adjusting the parameters in our HyGrasPr on the nurse rostering problem above show that $\alpha = 0.3$ and $N = 15$ can make our HyGrasPr perform well. Then we compare our hyper heuristic with an existing algorithm on the instances.

We choose 10 typical problem instances from the nurse rostering problem instances. We make comparison with the Simulated Annealing Hyper-Heuristic (SAHH) (Bai, Blazewicz, Burke, Kendall, & McCollum, 2007) proposed by Bai et al. SAHH is a hyper heuristic which includes a stochastic heuristic selection mechanism, a simulated annealing acceptance criterion, and a short-term memory. SAHH learns priorities of different low level heuristics from their historical performance and starts to select them depending on priorities. Therefore, the heuristics with good performance have higher priorities and are more likely to be chosen. The simulated annealing acceptance criterion is used to determine whether to be applied the selected heuristic. Information about the acceptance decisions by the acceptance criterion is then fed back to the heuristic selection mechanism in order to make good better decisions in the future. In (Bai et al., 2007), it was shown that SAHH is a well performed hyper-heuristic. SAHH can also be used for the nurse rostering problem. The number of iterations of SAHH is set to $K = 12000$ as described in (Bai et al., 2007). On each instance, the running time of our HyGrasPr is limited to the used time of SAHH. In Table 7, we present the results for HyGrasPr, SAHH and HyGrasPr without Path-relinking.

From the result we can conclude that our hyper heuristic can outperform the existing algorithm, SAHH. Among the 10 instances, HyGrasPr can get 8 better solutions out of 10 instances. We also compare HyGrasPr with the hyper heuristic implemented by GRASP without path-relinking. The performance of HyGrasPr is much better than that of HyGrasPr without path-relinking. This result shows that the path-relinking mechanism is very useful in our framework. In the path-relinking phase, the process the guidance of the LLH sequence from the initial one to the guiding one can provide various LLH sequences as the candidate LLH sequences. The performance of HyGrasPr is also better than that of SAHH.

To show the details of our experiments, we analyze the solving procedure of two algorithms, SAHH and HyGrasPr. Figure 1 presents the details of the processes when solving the instance ORTEC02 using SAHH and HyGrasPr. The results show that our algorithm HyGrasPr can improve solution step by step while SAHH provides a jumping curve.

Figure 1. Solving procedures of SAHH and HyGrasPr on the ORTEC02 instance



At the same time for both the algorithms, the solutions in SAHH are not as better as those in HyGrasPr. Our HyGrasPr provides a much stronger searching ability.

CONCLUSION AND FUTURE WORK

In this paper, we propose a Hyper-heuristic using GRASP with Path-Relinking (HyGrasPr). The procedure of HyGrasPr consists of three phases, such as the construction phase, the local search phase, and the path-relinking phase. To present the experimental results, we give a case study on the nurse rostering problem. The results show that HyGrasPr can achieve better solutions on 8 out of 10 instances than the existing simulate annealing hyper-heuristic.

In future work, we plan to give an empirical study on the performance of our HyGrasPr. Since the mechanism behind HyGrasPr is a little time-consuming, we want to design some improvement to reduce the running time. Another future work is to apply our HyGrasPr to some other problems. For example, we plan to give a comparison among the family of the nurse rostering problem. This work can show the generalization ability of our work.

ACKNOWLEDGMENTS

Our work is partially supported by the Natural Science Foundation of China under Grant No. 60805024, the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20070141020.

REFERENCES

Bai, R., Blazewicz, J., Burke, E. K., Kendall, G., & McCollum, B. (2007). *A simulated annealing hyper-heuristic methodology for flexible decision support* (Tech. Rep. No. NOTTCS-TR-2007-8). Nottingham, UK: University of Nottingham, School of Computer Science and Information Technology.

Bai, R., Burke, E. K., & Kendall, G. (2008). Heuristic, meta-heuristic and hyper-heuristic approaches for fresh produce inventory control and shelf space allocation. *The Journal of the Operational Research Society*, *59*, 1387–1397. doi:.doi:10.1057/palgrave.jors.2602463

Bai, R., & Kendall, G. (2005). An investigation of automated planograms using a simulated annealing based hyper-heuristics. In T. Ibaraki, et al. (Eds.), *Metaheuristics: Progress as a real problem solver* (pp. 87–108). Berlin, Germany: Springer. doi:10.1007/0-387-25383-1_4

Burke, E. K., Causmaecker, P. D., Berghe, G. V., & Landeghem, H. V. (2004). The state of the art of nurse rostering. *Journal of Scheduling*, *7*(6), 441–499. doi:.doi:10.1023/B:JOSH.0000046076.75950.0b

Burke, E. K., Curtois, T., Qu, R., & Vanden-Berghe, G. (2008). *Problem model for nurse rostering benchmark instances.* Retrieved December 14, 2010, from http://www.cs.nott.ac.uk/~tec/NRP/papers/ANROM.pdf

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., & Qu, R. (2010). *Hyper-heuristics: a survey of the state of the art* (Tech. Rep. No. NOTTCS-TR-SUB-0906241418). Nottingham, UK: University of Nottingham, School of Computer Science and Information Technology.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. R. (2010). A classification of hyper-heuristic approaches. In M. Gendreau & J. Potvin (Eds.), *Handbook of metaheuristics* (2nd ed., pp. 449–468). Berlin, Germany: Springer. doi:10.1007/978-1-4419-1665-5_15

Burke, E. K., Kendall, G., Newall, J., Hart, E., Ross, P., & Schulenburg, S. (2003). Hyper-Heuristics: An emerging direction in modern search technology . In Glover, F., & Konchenberger, G. (Eds.), *Handbook of metaheuristics* (pp. 457–474). New York, NY: Kluwer Academic Publishers.

Burke, E. K., Kendall, G., Silva, D. L., O'Brien, R., & Soubeiga, E. (2005). An ant algorithm hyperheuristic for the project presentation scheduling problem. In *Proceedings of the IEEE Conference on Evolutionary Computation* (pp. 2263-2270). Washington, DC: IEEE Computer Society.

Burke, E. K., Kendall, G., & Soubeiga, E. (2003). A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, *9*(6), 451–490. doi:.doi:10.1023/B:HEUR.0000012446.94732.b6

Burke, E. K., Li, J., & Qu, R. (2010). 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. doi:.doi:10.1016/j.ejor.2009.07.036

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, *176*(1), 177–192. doi:.doi:10.1016/j.ejor.2005.08.012

Croes, G. A. (1958). A method for solving traveling salesman problem. *Operations Research*, *6*, 791–812. doi:.doi:10.1287/opre.6.6.791

Curtois, T., Ochoa, M., Hyde, M., & Vazquez-Rodriguez, J. A. (2010). *A HyFlex module for the personnel scheduling problem*. Nottingham, UK: University of Nottingham, School of Computer Science and Information Technology.

Dowsland, K. A., Soubeiga, E., & Burke, E. K. (2007). A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. *European Journal of Operational Research*, *179*(3), 759–774. doi:.doi:10.1016/j.ejor.2005.03.058

Festa, P., Pardalos, P. M., Pitsoulis, L. S., & Resende, M. G. C. (2006). GRASP with path relinking for the weighted MAXSAT problem. *Journal of Experimental Algorithmics*, *11*, 1–16.

Ochoa, G., Qu, R., & Burke, E. K. (2009). Analyzing the landscape of a graph based hyper-heuristic for timetabling problems. In F. Rothlauf (Ed.), *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (pp. 341-348). New York, NY: ACM.

Qu, R., & Burke, E. K. (2009). Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems. *The Journal of the Operational Research Society*, *60*(9), 1273–1285. doi:.doi:10.1057/jors.2008.102

Remde, S., Cowling, P. I., Dahal, K. P., & Colledge, N. (2006). Exact/heuristic hybrids using rVNS and hyperheuristics for workforce scheduling. In C. Cotta et al. (Eds.), *Evolutionary computation in combinatorial optimization* (LNCS 4446, pp. 188-197).

Remde, S., Dahal, K. P., Cowling, P. I., & Colledge, N. (2009). Binary exponential back off for tabu tenure in hyperheuristics. In C. Cotta et al. (Eds.), *Evolutionary computation in combinatorial optimization* (LNCS 5482, pp. 109-120).

Ren, Z., Jiang, H., Xuan, J., & Luo, Z. (2010). Ant based hyper heuristics with space reduction: A case study of the p-Median problem. In R. Schaefer et al. (Eds.), *Proceedings of the Parallel Problem Solving from Nature Conference (PPSN XI)* (LNCS 6238, pp. 546-555).

Resende, M. G. C., & Ribeiro, C. C. (2003). Greedy random adaptive search procedures . In Glover, F., & Konchenberger, G. (Eds.), *Handbook of metaheuristics* (pp. 219–251). New York, NY: Kluwer Academic Publishers.

Ross, P., Marin-Blazquez, J. G., Schulenburg, S., & Hart, E. (2003). Learning a procedure that can solve hard bin-packing problems: A new GA-Based approach to hyper-heuristics. In E. Cantú-Paz et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)* (LNCS 2724, pp. 1295-1306).

Terashima-Marin, H., Moran-Saavedra, A., & Ross, P. (2005). Forming hyper-heuristics with GAs when solving 2D-regular cutting stock problems. In *Proceedings of the IEEE Conference on Evolutionary Computation* (pp. 1104-1110). Washington, DC: IEEE Computer Society.

He Jiang is an associate professor and Ph.D. supervisor of School of Software, Dalian University of Technology. Dr. Jiang received the B.A. degree and the Ph.D. degree in computer science from University of Science & Technology of China, Hefei, China in 1999 and 2005, respectively. He conducts active research in various information technology related areas, including evolutionary computation, heuristics, combinatorial optimization, and data mining.

Junying Qiu is a master student of School of Software, Dalian University of Technology. He received the B.A. degree in software engineering from Dalian University of Technology, Dalian, China in 2009. His research interest is hyper-heuristics.

Jifeng Xuan is a Ph.D. candidate of School of Mathematical Sciences, Dalian University of Technology. He received the B.A. degree in software engineering from Dalian University of Technology, Dalian, China in 2007. His research interests include heuristics and software maintenance.