EVOLUTIONARY RUIN AND STOCHASTIC RECREATE: A CASE STUDY ON THE EXAM TIMETABLING PROBLEM

Jingpeng Li
Division of Computer Science
The University of Nottingham Ningbo China
Ningbo 315100, China
E-mail: Jingpeng.Li@nottingham.edu.cn

Rong Qu
School of Computer Science
The University of Nottingham
Nottingham NG8 1BB, United Kingdom
E-mail: rxq@cs.nott.ac.uk

KEYWORDS

Evolutionary algorithm, Intelligent machine learning system, Combinatorial optimisation, Educational timetabling.

ABSTRACT

This paper presents a new class of intelligent systems, called Evolutionary Ruin and Stochastic Recreate, that can learn and adapt to the changing enviroment. It improves the original Ruin and Recreate principle's performance by incorporating an *Evolutionary Ruin* step which implements evolution within a single solution. In the proposed approach, a cycle of *Solution Decomposition*, *Evolutionary Ruin* and *Stochastic Recreate* continues until stopping conditions are reached. The *Solution Decomposition* step first uses some domain knowledge to break a solution down into its components and assign a score to each. The *Evolutionary Ruin* step then applies two operators (namely *Selection* and *Mutation*) to destroy a certain fraction of the entire solution. After the above steps, an input solution becomes partial and thus the resulting partial solution needs to be repaired. The repair is carried out by using the *Stochastic Recreate* step to reintroduce the removed items in a specific way (somewhat stochastic in order to have a better chance to jump out of the local optima), and then ask the underlying improvement heuristic whether this move will be accepted. These three steps are executed in sequence until a specific stopping condition is reached. Therefore, optimisation is achieved by solution disruption, iterative improvement and a stochastic constructive repair process performed within. Encouraging experimental results on exam timetabling problems are reported.

1 INTRODUCTION

Exam timetabling can be considered as the process of assigning a set of events (i.e. exams) into a limited number of timeslots subject to a set of constraints. The problem has attracted a significant level of research interest since the 1960's. The general timetabling problem comes in many different guises such as nurse rostering (Cheang et al. 2003; Li et al. 2012), sports timetabling (Easton et al. 2004), transportation timetabling (Kwan 2004) and educational timetabling (Carter and Laporte 1996; Schaerf 1999; Petrovic and Burke 2004; Qu et al. 2009). Educational timetabling problems are prob-ably the most widely studied.

Since exam timetabling problems are general NP-hard combinatorial problems which are unlikely to be solved optimally in polynomial time, various methods such as local search-based heuristics (Casey and Thompson 2004; Burke et al. 2004; Burke and Newall 2004), knowledge-based systems (Burke et al. 2006) and hyper-heuristics (Qu and Burke 2009) have been studied. Over the past dec-ade, meta-heuristics have attracted the most attention, including genetic algorithms (Erben 2001; Paquete and Fonseca 2001), ant algorithms (Dowsland and Thompson 2005), tabu search (Burke et al. 2003; White et al. 2004), simulated annealing (Thompson and Dowsland 1998). A number of at-tempts have also been made using other meta-heuristics (Burke et al. 1996; Duong and Lam 2004). The methods and techniques that have been used over the years to tackle exam timetabling problems have tended to draw on problem-specific information and particular heuristics. In this paper, we are trying to deal with the goal of developing a new class of search systems. We use the well-studied exam timetable problem as the test bed.

The work that is presented here is based on the observation that, in most real world problems, the solutions consist of components which are intricately woven together. Each solution component may be a strong candidate in its own right, but it also has to fit well with other components in the enviroment. To deal with these components, Schrimpf et al. (2000) proposed a technique called Ruin and Recreate (R&R) principal, and claimed it could be a general approach for various combinatorial optimisation problems. In this paper, we further extend the idea by incorporating some evolutionary features into the searching process. We term the enhanced version Evolutionary Ruin and Stochastic Recreate (ER&SR). Its general idea is to break a solution down into its components and assign a score to each by an evaluation function working under dynamic environments. The scores are employed as fitness values which determine the chances for the components to survive in the current solution.

2 A GENERAL DESCRIPTION OF THE ER&SR

The Ruin and Recreate (R&R) method uses the concepts of simulated annealing (Kirkpatrick 1983) or threshold accepting (Dueck and Scheuer 1990) with large moves instead of smaller ones. For simple structured problems like the traveling salesman problem, the need of using large moves is not obvi-ous, because algorithms usually generate near optimal solutions with very small moves already. However, for complex problems like exam timetabling problems, difficulties arise if still using such small moves, because complex problems can often be seen as discontinuous: if walking one step from a solution to a neighbouring solution, the qualities of new solutions may be dramatically different, i.e., the landscapes of these problem areas can be very rugged.

Solutions of complex problems usually have many soft and/or hard constraints, which makes it dif-ficult to get just feasible solutions. Neighbouring solutions of complex schedules, for instance, are usually infeasible solutions. It may be very hard to walk in such a complex landscape from one feasi-ble solution to another neighboured feasible solution. The common method of avoiding the infeasibil-ity problem for many forms of the classical algorithms is to impose artificial penalty functions, but this method would typically make the algorithms get stuck in slightly infeasible solutions which might not be allowed at all.

Naturally, one will think in a different paradigm: ruin and recreate. We ruin a quite large portion of the solution and try to rebuild the solution as best as we can, with the hope that the new solution is better than the previous one. The R&R approach is just based on the above idea which has shown an important advantage: if destroying a large part of the previous solution, we have more freedom to generate a new one, and thus it is more likely to find again a feasible solution in this larger solution space. Hence, it is reasonable to believe that problems with many side conditions, or with very com-plex objective functions, are more tractable using special large moves.

Based on the general R&R principal, this paper presents a more advanced technique called ER&SR which has never been reported in the literature before. The new technique applies two operators of *Selection* and *Mutation* as the ruining strategies, trying to mimic the evolution on single solutions. Each component in the solution has to continuously demonstrate its worthiness to stay in the solution. Hence in each iteration, a number of components will be deemed not worth keeping. The evolutionary strategy adopted may also throw out, with a low probability, some worthy components. Any destroyed component is then reintroduced by using a specific algorithm. Of key importance is that the admittance of a new component is determined by a dynamic evaluation function, which takes into account of how well the prospective component will fit in with others already in the solution. The above processes are iterated together with the remainder of the classical R&R. Thus the global optimisation procedure is based on solution disruption and iterative improvement, while a reconstructive process is performed within.

As outlined, our proposed ER&SR algorithm consists of the following three parts: *Solution Decomposition*, *Evolutionary Ruin* and *Stochastic Recreate*. It executes these parts in a loop on one solu-tion until a stopping condition reached. The first part of *Solution Decomposition* is based on the observation that in most real world combinatorial optimization problems, the solutions consist of components which are intricately woven together in a nonlinear, non-additive fashion. Each solution component may be a strong candidate in its own right, but it also has to fit well with other components. Its general idea is to use some expert's domain knowledge to break a solution down into its components and assign a score to each. The higher the score, the fitter the related component is.

The second part of *Evolutionary Ruin* is based on the consideration that the incumbent solution must be changed not only locally but also over a macroscopic scale, depending on the solution composition defined by the proceeding *Solution Decomposition* part. This part applies two operators of *Selection* and *Mutation* to destroy a certain fraction of the entire solution. The *Selection* operator removes some components based on Darwin's survival of the fitness mechanism, while the *Mutation* operator further removes some components in a totally random manner. Hence, the destroyed part of the solution would sometimes be large enough such that the impact of the "bomb" that is thrown on the solution will be noticeable not only locally but in the whole system. On the other hand, the destroyed part would sometimes be small enough so that at least a main portion of the solution (i.e. a skeleton) remains to facilitate the next solution rebuild.

The third part of *Stochastic Recreate* follows to reintroduce the removed items in a specific way (somewhat stochastic in order to have a better chance to jump out of the local optima), and then ask the underlying improvement heuristic (e.g. hill-climbing, simulated annealing, or great deluge) whether this move will be accepted. These three parts are executed in sequence until a specific stopping condition is reached.

The above mentioned model is a general framework and many well-known search methods belong to its special case. For example, assume at each iteration x components are removed from an n-component solution, and let p be the acceptance criterion.

- If $x = 0$, then it is a non-iterative method as only one single solution will be generated;
- If $x \leq 3$, then it is a local search method that uses small moves to change the configurations;

- If $x = n$, then it is a constructive method with random starting points;
- If (no *Solution Decomposition*) & (no *Selection*), then it equals to the R&R principle.
- If (p = "hill-climbing") & (no *Solution Decomposition*) & (no *Selection*), then is equals to the most basic evolutionary algorithm called "(1+1)EA", in which its population is composed by two individuals only: one being the parent, the other the offspring.

3 EXAM TIMETABLING

Exam timetabling can be considered to be the process of assigning a set of events (i.e. exams) into a limited number of timeslots subject to a set of constraints. Constraints are usually divided into two types: hard and soft. A hard constraint cannot be violated under any circumstances. A typical example is two exams with common students involved cannot be scheduled into the same timeslot. A soft constraint is one that should be satisfied if possible but its satisfaction is not essential. A typical example is exams taken by common students should be spread out over the available timeslots so that students do not have to sit two exams that are too close to each other. Solutions with no violations of hard constraints are called feasible solutions. How much the soft constraints are satisfied gives an indication of how good the solutions (timetables) are.

In a simplified timetabling problem, if we are only concerned with hard constraints, the problem can be represented by a graph colouring model. Vertices in the graph represent exams in the problem, and edges representing the conflicts between exams (i.e. with common students). The problem is to minimise the colours used to colour all vertices, while avoiding the assignment of two adjacent vertices to the same colour. Graph colouring problems are among the most important problems in graph theory and are known to be NP-hard (Karp 1972).

The following objective function is used in (Burke et al. 2007) and many other papers in the literature to calculate the cost of an obtained feasible solution x:

$$\text{Min} \quad f(x) = \sum_{k=1}^{m-1} \sum_{l=k+1}^{m} (w_i \times s_{kl}) \Big/ S, \; i \in \{0,1,2,3,4\}, \quad (1)$$

where

s_{kl} is the number of students involved in both exams e_k and e_l, if $i = |t_l - t_k| < 5$;
$w_i = 2^{4-i}$ is the cost of assigning two conflicted exams e_k and e_l with i timeslots apart, if $i = |t_l - t_k| < 5$ (i.e. $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$, $w_5 = 1$; t_l and t_k as the timeslots of e_l and e_k, respectively);
m is the number of exams in the problem;
S is the number of students in the problem.

4 ER&SR FOR EXAM TIMETABLING

This section presents an ER&SR for exam timetabling. Starting from a randomly generated initial timetable, the steps described in section 4.1 to 4.3 are executed in sequence in a loop until a user specified parameter (e.g. CPU-time or solution quality) is reached or no improvement has been achieved for a certain number of iterations. During each iteration, an unfit portion of the working timetable is removed. Broken timetables are repaired by the constructing heuristic. Throughout the iterations, the best is retained and finally returned as the preserved timetable.

4.1 Solution Decomposition

This step is to evaluate the current arrangement for each event $e_k, k \in \{1,...,m\}$, in a timetable. In this step, the fitness of each event for a generated timetable is computed. The purpose of computing this measure is to determine which events are in positions that contribute more towards the cost reduction for the resulting solution. We can formulate a normalized evaluation function $F_t(e_k), k \in \{1,...,m\}$, at the t-th iteration as

$$F_t(e_k) = \frac{\max(C_t(e_1),...,C_t(e_m)) - C_t(e_k)}{\max(C_t(e_1),...,C_t(e_m)) - \min(C_t(e_1),...,C_t(e_m))}, \quad (2)$$

and

$$C_t(e_k) = \sum_{l=1}^{k-1} (w_i \times s_{kl}) + \sum_{l=k+1}^{m} (w_i \times s_{kl}), \; i \in \{0,1,2,3,4\}, \quad (3)$$

where $C_t(e_k)$ is the cost value brought by event e_k, and w_i uses the same definition as in Equation (1).

4.2 Evolutionary Ruin

This step is to decide whether a component (i.e. an event $e_k, k \in \{1,...,m\}$) in a current timetable should be retained or discarded. The decision is made by implementing two operators of *Selection* and *Mutation*. The *Selection* operator compares its fitness value $F_t(e_k)$ to a random number $p_s^{(t)}$ generated for each iteration t in the range [0, 1]. If $F_t(e_k) \geq p_s^{(t)}$, then e_k will remain in its present allocation, otherwise e_k will be removed from the current timetable. By using *Selection*, an event e_k with larger fitness value $F_t(e_k)$ has a higher probability to survive in the current timetable. The *Mutation* operator follows to mutate the retained events e_k, i.e. randomly discarding them from the partial timetable at a small rate $p_m^{(t)}$. Compared with the selection rate $p_s^{(t)}$ which is randomly generated for each iteration t, the mutation rate $p_m^{(t)}$ should be much smaller to aid convergence.

4.3 Stochastic Recreate

The *Stochastic Recreate* task is to rebuild a partial timetable by assigning unscheduled events to available

timeslots. Once a specific event has been determined, the following two steps will be executed: Step 1 finds all its available timeslots without any conflict exams; Step 2 chooses the timeslot with the smallest increase on the overall cost defined by Equation (1).

Based on the domain knowledge of timetabling, there are many heuristics that can be used to determine the order for the events to be rescheduled. Here we use the following four graph-based heuristics reported in (Qu and Burke 2009): largest degree first (H_1), largest weighted degree first (H_2), largest color degree first (H_3) and least saturation degree first (H_4). Let $p_1^{(k)}$, $p_2^{(k)}$, $p_3^{(k)}$ and $p_4^{(k)}$ be the probabilities of using heuristics H_1, H_2, H_3 and H_4 respectively for the rescheduling of event e_k. These heuristics are alternatively used in each step of recreate, satisfying $\sum_{j=1}^{4} p_j^{(k)} = 1$.

The H_1 heuristic orders events in descending order by the number of conflicts they have with other exams. This heuristic aims to schedule first those events which have the most conflicts. It first goes through the conflict values (which are precalculated) for the unscheduled events, and then proceeds to Steps 1 and 2.

The H_2 heuristic orders events in descending order by the number of conflicts, each of which is weighted by the number of students involved. Among events with the same degree, this heuristic gives higher priority to those with a larger number of students involved. Like H_1, it first goes through the weighted conflict values for the unscheduled exams, and then proceeds to Steps 1 and 2.

The H_3 heuristic orders events in a descending order in terms of the number of conflicts with the other events that have already been placed in the timetable. The degrees of the events not yet scheduled are changed according to the situations encountered at each step of the solution construction. Unlike H_1 and H_2, the conflict value for each exam needs to be firstly updated before scheduling an exam. This involves updating an ancillary matrix that contains the conflict values between any pairs of an unscheduled exam and a scheduled exam. H_3 then goes through the new conflict values for the unscheduled exams, and next proceeds to Steps 1 and 2.

The H_4 heuristic orders events in ascending order in terms of the number of available timeslots that can be selected without violating hard constraints. The priorities of events to be ordered and scheduled are changed dynamically as the solution is constructed. The number of available timeslots for each event needs to be firstly calculated before rescheduling an event. This process can be regarded as executing Step 1 repeatedly for all unscheduled exams. H_4 then goes through the saturation degree numbers for all the unscheduled events, and next proceeds to Steps 1 and 2.

5 EXPERIMENTAL RESULTS

The exam timetabling problems we tested in this paper were first introduced in (Carter et al. 1996), and have been widely tested by a number of approaches during the last ten years. The dataset consists of 13 problems from different institutions, among which 11 have been more heavily investigated because of errors in the other two problems. A more detailed discussion of those datasets (and the difficulties caused by different instances circulating under the same name) was given in (Qu et al. 2009).Our aim with these experiments is not to beat the state-of-the-art approaches in the literature (although the results are competitive with the best results reported), but to present the potential of this more generic methodology to be easily employed and to perform adaptively on a range of different timetabling or optimisation problems.

Table 1 presents the characteristics of the 11 problems in the dataset. Rows 2-5 include the number of exams, the number of students, the number of available time slots and the problem density. The problem size ranges from 81 to 682 exams, from 611 to 18416 students and from 10 to 35 time slots. The "density" (ranged from 0.06 to 0.42) gives the conflict density of elements with value 1 in the conflict matrix, where element $C_{ij} = 1$ if events i and j conflict, $C_{ij} = 0$ otherwise. More details about the benchmark dataset can be found at http://www.asap.cs.nott.ac.uk/resources/data.shtml.

Table 1: Characteristics of the Benchmark Problems

	car91	car92	ear83	hec92	kfu93	lse91	sta83	tre92	uta93	ute92	york83
Exams	682	543	190	81	461	381	139	261	622	184	181
Students	16925	18419	1125	2823	5349	2726	611	4360	21266	2750	941
Timeslots	35	32	24	18	20	18	13	23	35	10	21
Density	0.13	0.14	0.27	0.42	0.06	0.06	0.14	0.18	0.13	0.08	0.29

Table 2 presents the 20 runs' results on the benchmark exam timetabling problems of the original R&R and the enhanced ER&SR. For comparison, it also lists the results of the state-of-the-art approaches in the literature. The best results among all of the approaches are highlighted. Both algorithms were coded in C++ and

implemented on an Intel Core 2 Duo 1.86GHz machine with 2.0GB of RAM under Window XP. The stopping condition is no improvement has been made after 1000 iterations. We can see that the ER&SR outperforms the R&R over all of the problems in terms of best and average results. It is also very consistent on all of the runs with distinct random seeds.

Table 2: Comparison Results on Benchmark Problems.

	car91	car92	ear83	hec92	kfu93	lse91	sta83	tre92	uta93	ute92	york83
R&R best	5.4	4.9	38.6	12.2	15.3	12.8	161.2	8.8	3.6	30.1	40.7
R&R avg	6.1	5.3	39.9	12.5	15.6	13.2	163.6	9.2	4.2	31.3	43.2
ER&SR best	5.3	4.7	29.7	10.1	13.7	10.2	157.3	8.4	3.3	25.3	37.8
ER&SR avg	5.4	5.2	30.0	11.9	14.1	10.5	157.9	8.7	3.5	26.6	39.0
Abdullah et al. 2007	5.2	4.4	34.9	10.3	**13.5**	10.2	159.2	8.7	3.6	26.0	**36.2**
Asmuni et al. 2005	5.2	4.5	37.0	11.8	15.8	12.1	160.4	8.7	3.6	27.8	40.7
Burke & Newall, 2004	**4.6**	**4.0**	37.1	11.5	13.9	10.8	168.7	8.4	**3.2**	25.8	36.8
Burke et al. 2004	4.8	4.2	35.4	10.8	13.7	10.4	159.1	**8.3**	3.4	25.7	36.7
Caramia et al. 1982	6.6	6.0	**29.3**	**9.2**	13.8	**9.6**	158.2	9.4	3.5	**24.4**	36.2
Carter et al. 1996	7.1	6.2	36.4	10.8	14.0	10.5	161.5	9.6	3.5	25.8	41.7
Gaspero & Schaerf 2001	6.2	5.2	45.7	12.4	18.0	15.5	160.8	10.0	4.2	29.0	42.0
Merlot et al. 2002	5.1	4.3	35.1	10.6	13.5	10.5	**157.3**	8.4	3.5	25.1	37.4

It can be observed that the best results reported in the literature were obtained by different approaches over the years. Among the 8 approaches compared (which have obtained the best results in the literature on the benchmarks), our ER&SR obtained competitive results. However, the most important point to make here is that all of the other approaches were specifically designed for the exam timetabling problem.

6 CONCLUSIONS

This paper presents a new approach to solve timetabling problems based on the original idea of R&R, by incorporating two operators of *Selection* and *Mutation* in its *Evolutionary Ruin* step. In our proposed ERSR, a cycle of *Solution Decomposition*, *Evolutionary Ruin* and *Stochastic Recreate* continues until stopping conditions are reached. Taken as a whole, the ER&SR implements evolution within a single solution and carries out search by solution disruption, iterative improvement and a stochastic constructive process. The experiments have demonstrated that the proposed approach performs very efficiently and competitively.

The architecture of the ER&SR is innovative, and thus there is still some room for further improvement. In the *Solution Decomposition* part, we will study the formulation of domain knowledge for different types of other problems and the influence of different evaluation rules. In the *Evolutionary Ruin* part, we will study the suitable range for the number of components to be destroyed and the condition of applying a large move or a small move. For the *Stochastic Recreate* part, we will study the types of reconstruction methods that are unsuited for generating optimum or near-optimum results. Furthermore, we will evaluate the proposed algorithm with different operators applied in its three parts and the best combination we should try.

REFERENCES

Abdullah S.; S. Ahmadi; E.K. Burke; and M. Dror. 2007. "Investigating Ahuja-Orlin's Large Neighbourhood Search for Examination Timetabling." *OR Spectrum* 29, 351-372.

Asmuni H.; E.K. Burke; and J. Garibaldi. 2005. "*Fuzzy Multiple Ordering Criteria for Examination Timetabling*." *Practice and Theory of Automated Timetabling*. Springer Lecture Notes in Computer Science 3616, 334-353.

Burke E.K.; Y. Bykov; J.P. Newall; and S. Petrovic. 2004. "A Time-Predefined Local Search Approach to Exam Timetabling Problems." *IIE Transactions* 36, 509-528.

Burke E.K.; G. Kendall; and E. Soubeiga. 2003. "A Tabu-search Hyperheuristic for Timetabling and Rostering." *Journal of Heuristics* 9, 451-470.

Burke E.K.; B. McCollum; A. Meisel; S. Petrovic; and R. Qu. 2007. "*A Graph-based Hyper-heuristic for Educational Timetabling Problems*." *European Journal of Operational Research* 176, 177-192.

Burke E.K. and J.P. Newall. 2004. "Enhancing Timetable Solutions with Local Search Methods." *Practice and Theory of Automated Timetabling*. Springer Lecture Notes in Computer Science 2740, 195-206.

Burke E.K. and J. Newall. 2004. "Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings." *Annals of operations Research* 129, 107-134.

Burke E.K.; J.P. Newalland R.F. Weare. 1996. "A Memetic Algorithm for University Exam Timetabling." *Practice and Theory of Automated Timetabling*. Springer Lecture Notes in Computer Science 1153, 241-250.

Burke E.K.; S. Petrovic; and R. Qu. 2006. "Case Based Heuristic Selection for Timetabling Problems," *Journal of Scheduling* 9, 115-132.

Carter M.; G. Laporte; and S. Lee. 1996. "Examination Timetabling: Algorithmic Strategies and Applications." *Journal of Operations Research Society* 47, 373-383.

Caramia M.; P. DellOlmo; and G. Italiano. 2001. "*New Algorithms for Examination Timetabling*." *Algorithm*

Engineering. Springer Lecture Notes in Computer Science 1982, 230-241.

Casey S. and J. Thompson. 2004. "GRASPing the Examination Scheduling Problem." *Practice and Theory of Automated Timetabling*. Springer Lecture Notes in Computer Science 2740, 232-244.

Cheang B.; H. Li; A. Lim; and B. Rodrigues. 2003. "Nurse Rostering Problems: a Bibliographic Survey." *European Journal of Operational Research* 151, 447-460.

Dueck G. and T. Scheuer. 1990. "Threshold Accepting: a General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing." *Journal of Computational Physics* 90, 161-175.

Duong T.A. and K.H. Lam. 2004. "Combining Constraint Programming and Simulated Annealing on University Exam Timetabling." In *Proceedings of RIVF 2004 Conference*, 205-210.

Dowsland K. and J. Thompson. 2005. "Ant Colony Optimization for the Examination Scheduling Problem." *Journal of Operations Research Society* 56, 426-438.

Easton K.; G. Nemhauser; and M. Trick. 2004. "Sports Scheduling," In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J. Leung (ed.). Chapter 52, CRC Press.

Erben W. 2001. "A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling." *Practice and Theory of Automated Timetabling*. Springer Lecture Notes in Computer Science 2079, 132-156.

Di Gaspero L. and A. Schaerf. 2001. "Tabu Search Techniques for Examination Timetabling." *Practice and Theory of Automated Timetabling*. Springer Lecture Notes in Computer Science 2079, 104-117.

Karp R.M. 1983. "Reducibility Among Combinatorial Problems." *Complexity of Computer Computations* 4, 85-103.

Kirkpatrick S.; C.D. Gelatt; and M.P. Vecchi. 1983. "Optimization by Simulated Annealing." Science 220, 671-680.

Kwan R.S.K. 2004. "Bus and Train Driver Scheduling." *Handbook of scheduling: Algorithms, Models, and Performance Analysis*, Chapter 51. CRC Press.

Li J.; E.K. Burke; T. Curtois; S. Petrovic; and R. Qu. 2012. "The Falling Tide Algorithm: a New Multi-objective Approach for Complex Workforce Scheduling." *OMEGA – The International Journal of Management Science* 40, 283-293.

Merlot L.; N. Boland; B. Hughes; and P. Stuckey. 2002. "A Hybrid Algorithm for the Examination Timetabling Problem." *Practice and Theory of Automated Timetabling*. Springer Lecture Notes in Computer Science 2740, 207-231.

Paquete L. and C.M. Fonseca. 2001. "A Study of Examination Timetabling with Multiobjective Evolutionary Algorithm." In *Proceedings of the 4th Meta-heuristics International Conference* (MIC 2001), 149-154.

Petrovic S. and E.K. Burke. 2004. "University Timetabling." *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapter 45, CRC Press.

Qu R.; E.K. Burke; B. McCollum; L.T.G. Merlot; and S.Y Lee. 2009. "A Survey of Search Methodologies and Automated System Development for Examination Timetabling." *Journal of Scheduling*12, 5-89.

Qu R. and E.K. Burke. 2009. "Hybridizations Within a Graph based Hyper-heuristic Framework for University Timetabling Problems." *Journal of the Operational Research Society* 60, 1273-1285.

Schaerf A. 1999. "A Survey of Automated Timetabling." *Artificial Intelligence Review* 13, 87-127.

Schrimpf G.; J. Schneider; H. Stamm-Wilbrand; and G. Dueck. 2000. "Record Breaking Optimization Results using the Ruin and Recreate Principle." *Journal of Computational Physics* 159, 139-171.

Thompson J. and K. Dowsland, 1998. "A Robust Simulated Annealing based Examination Timetabling System." *Computer & Operations Research* 25, 637-648.

White G.M.; B.S. Xie; and S. Zonjic. 2004. "Using Tabu Search with Longer-Term Memory and Relaxation to Create Examination Timetables." *European Journal of Operational Research* 153, 80-91.

AUTHOR BIOGRAPHIES

JINGPENG LI received the M.Sc. degree in computational mathematics from Huazhong University of Science and Technology, China, in 1998, and the Ph.D. degree in computer science from University of Leeds, Leeds, U.K., in 2002. He joined the School of Informatics, University of Bradford, U.K., as a Research Associate in 2003. Since 2004, he has been with the School of Computer Science, University of Nottingham (UK campus) as a Research Fellow initially, a permanent Senior Research Fellow later, and currently an Assistant Professor at the University's China campus. His research areas include Intelligent Transport Scheduling, Metaheuristics, Multi-Objective Decision Making, Optimization & Search Methodologies, Machine Learning, Data Mining, Markov Chain Analysis, Fuzzy Logic, and Image Process. He has published over 30 peer-reviewed research papers in a wide variety of world's leading journals and conference proceedings. Dr. Li has worked on five U.K. government-funded EPSRC projects and three Chinese government-funded NNSFC research projects covering the topics of human scheduling, next generation decision support, novel research directions in personnel rostering, general optimization systems, theoretical understanding of heuristics, ands public transport scheduling. In terms of academic services, he is a member of the Editorial Board of Wireless Sensor Network. In addition,.he is acting as a referee for more than 20 leading journals, and has served on the program committees for many international conferences. His e-mail address is: Jingpeng.Li@nottingham.edu.cn and his Web-page can be found at http://www.nott.ac.uk/~jpl.

RONG QU received the BSc degree in computer science from XiDian University, Xi'an, China, in 1996, and the Ph.D. degree in computer science from University of Nottingham, Nottingham, U.K., in 2002. She has been working in the School of Computer Science, University of Nottingham as a Lecturer since 2005. Her research areas include Metaheuristics, Constraint Programming, Integer Programming, Data Mining, and Knowledge Based Systems. She has published over 60 peer-reviewed research papers in a wide variety of world's leading journals and conference proceedings. Her e-mail address is: rxq@cs.nott.ac.uk

and her Web-page can be found at http://www.nott.ac.uk/~rxq.