Solving the Examination Timetabling Problem with the Shuffled Frog-leaping Algorithm

Nuno Leite[1,2,*], Fernando Melício[1,2] and Agostinho Rosa[2,3,†]

[1]*ISEL - Lisbon Polytechnic Institute, Rua Conselheiro Emídio Navarro, 1, 1959-007 Lisboa, Portugal*

[2]*LaSEEB - System and Robotics Institute, Av. Rovisco Pais 1, TN 6.21, 1049-001 Lisboa, Portugal*

[3]*Department of Bioengineering/IST, TU-Lisbon, Av. Rovisco Pais, 1, 1049-001 Lisboa, Portugal*

Keywords: Examination Timetabling Problem, Toronto Benchmarks, Memetic Algorithm, Shuffled Frog-Leaping Algorithm.

Abstract: Shuffled Frog-Leaping Algorithm (SFLA) is a recently proposed memetic metaheuristic algorithm for solving combinatorial optimisation problems. SFLA has both global and local search capabilities, and great convergence speed towards the global optimum. Compared to a genetic algorithm, the experimental results show an effective reduction of the number of evaluations required to find the global optimal solution. The Examination Timetabling Problem (ETTP) is a complex combinatorial optimisation problem faced by schools and universities every epoch. In this work, we apply the Shuffled Frog-Leaping Algorithm to solve the ETTP. The evolution step of the algorithm, specifically the local exploration in the submemeplex is adapted based on the standard SFLA. The algorithm was evaluated on the Toronto benchmark instances, and the preliminary experimental results obtained are comparable to those produced by state of art algorithms while requiring much less time.

1 INTRODUCTION

Examination timetabling is an important practical problem faced by schools and universities every epoch. This problem, termed *ETTP – Examination Timetabling Problem*, consists in scheduling exams into a limited number of time slots and rooms subject to a set of constraints, providing that no students should attend two or more exams at the same time. If the room capacity is infinite, the ETTP is classified as Uncapacitated ETTP, otherwise it is named Capacitated ETTP.

The ETTP belongs to the general class of timetabling problems that includes other educational timetabling problems (e.g. school and course timetabling), employee rostering, sports timetabling and others. These problems, in general, belong to the class of NP-complete problems, limiting the application of optimal deterministic algorithms (e.g. Mathematical programming techniques or Dynamic Programming) to simplified and small size problem instances. In approaching real timetabling problems, researchers often have investigated characteristics of the problem that can be exploited in order to apply a heuristic method capable of producing satisfactory results. Proposed methods that combine algorithmic strategies originating from the Operations Research and Artificial Intelligence present the state-of-the-art results.

Several approaches to solve the ETTP have been proposed since the first works in the 1960's decade. In its survey, Carter (Carter, 1986) classified these approaches into four types: sequential methods, cluster methods, constraint-based methods and generalised search. Petrovic and Burke (Petrovic and Burke, 2004) later added more six categories: hybrid evolutionary algorithms, metaheuristics, multi-criteria approaches, case based reasoning techniques, hyper-heuristics and adaptive approaches. A recent and detailed overview of the proposed methods to solve the ETTP can be found in (Qu et al., 2009).

The Shuffled Frog-Leaping Algorithm (SFLA) is a memetic metaheuristic proposed in 2003 (Eusuff and Lansey, 2003) (Eusuff et al., 2006). The SFLA was applied to many areas and problems, namely: solving TSP (Xue-hui et al., 2008), Cluster-

*Nuno's work was partially supported by the FCT SFRH/PROTEC/67953/2010 grant.

†This work was also supported by the FCT Project PEst-OE/EEI/LA0009/2011.

Leite N., Melício F. and Rosa A..
Solving the Examination Timetabling Problem with the Shuffled Frog-leaping Algorithm.
DOI: 10.5220/0004636001750180
In *Proceedings of the 5th International Joint Conference on Computational Intelligence* (ECTA-2013), pages 175-180
ISBN: 978-989-8565-77-8

ing (Amiri et al., 2009), Flow-shop Scheduling (Xu et al., 2013), multiobjective optimization (Rahimi-Vahed and Mirzaei, 2007), and others.

To the best of our knowledge, the application of SFLA to the ETTP include only the work of Wang et al. (Wang et al., 2009) (in chinese). The authors propose a Discrete SFLA (DSFLA) and apply it to the ETTP. Solutions are encoded using a time permutation scheme suited to be manipulated by the DSFLA. The algorithm is evaluated on four datasets of the Capacited Toronto benchmark (Toronto variant *c* in (Qu et al., 2009)). In our work we present a novel application of SFLA to the ETTP. Specifically, we adapt the SFLA model by proposing a new evolution operator which is capable of exploring the search space conveniently. The proposed algorithm is applied to the complete set of Uncapacitated Toronto benchmark data (Toronto variant *b* in (Qu et al., 2009)), and compared with other techniques in the literature. The original version (version I (Carter et al., 1996)) of the datasets was evaluated. The ETTP mathematical specification can be found in (Abdullah et al., 2010).

The rest of the paper is organized as follows. The next section presents the mathematical model of the SFLA followed by the adaptation to the ETTP in Section 3. Experimental results of comparing the adapted SFLA with other algorithms from the literature are reported and discussed in Section 4. Finally, Section 5 present conclusions of the research undertaken and discussion on the future work.

2 SFLA MODEL

In the SFLA, a population of F frogs, denoted $U(i)$, $i = 1,\ldots,F$, with identical structure, but different adaptation to the environment, is maintained. The F frogs are divided into m substructures called memeplexes, where they "search for food" (they are optimized, in the algorithm sense) and meanwhile, exchange information (exchange memes) with other frogs, trying to reach the food localisation (global optimum). Each memeplex comprise n frogs, so that $F = mn$. After searching locally in their memeplex, the frogs are ranked and shuffled in order to go, eventually, to a different memeplex and exchange their memes with the frogs located there (Figure 1). The ranking comprises sorting the frogs in descending order of performance. The partition of frogs is as follows. The first frog (the frog with the best fitness) in the sorted list is allocated to the memeplex 1; the second frog is allocated to the memeplex 2, and so on, so that the frog m will go to memeplex m; then, the $m+1$ frog will go to memeplex 1 again, and the process is repeated for the remainder frogs. The local search employed corresponds to the so called Frog-Leaping local search (Figure 2).

The ith frog fitness is denoted by $f(i)$. To prevent being trapped in a local optimum, a submemeplex of size $q < n$ is constructed in each memeplex, which consists of frogs selected according to their performance. For each submemeplex, P_b and P_w denote, respectively, the best and worst frog. In the end of each iteration of the Frog-Leaping local search, the worst frog in the submemeplex is updated according to the following rule:

$$S = \begin{cases} \min\{\text{int}[rand * (P_b - P_w)], S_{max}\}, \\ \qquad \text{for a positive step} \\ \max\{\text{int}[rand * (P_b - P_w)], -S_{max}\}, \\ \qquad \text{for a negative step} \end{cases} \quad (1)$$

$$U(q) = P_w + S \quad (2)$$

where S denotes the updated step size of the shuffled frog-leaping, *rand* represents a random number between $(0,1)$ and S_{max} is defined as the maximum step size that any frog can take. The idea of this step is to update the worst frog position towards the direction of the best frog in the memeplex (or towards the direction of the global best frog, as indicated in Figure 2).

3 APPLYING SFLA TO ETTP

In this section we describe how the SFLA was adapted in order to solve the ETTP.

3.1 Solution Representation

Each individual frog (solution) is represented by a vector of dimension equal to the number of periods, where each position contains the list of exams scheduled at that period. Figure 3 shows the graphical representation of three solutions (the t_i's are the time slots and the e_j's are the exams).

3.2 Initialisation of the Frog Population

The initial frog population is created using a construction algorithm which is based on the saturation degree graph colouring heuristic (Carter and Laporte, 1996). To construct each solution, the approach begins with an empty timetable and the most difficult exams to insert (exams with the least number of available periods) should be selected next for insertion. The remainder exams to be inserted have more feasible available periods, and so they are more easy to insert. In this heuristic only the hard constraints are met

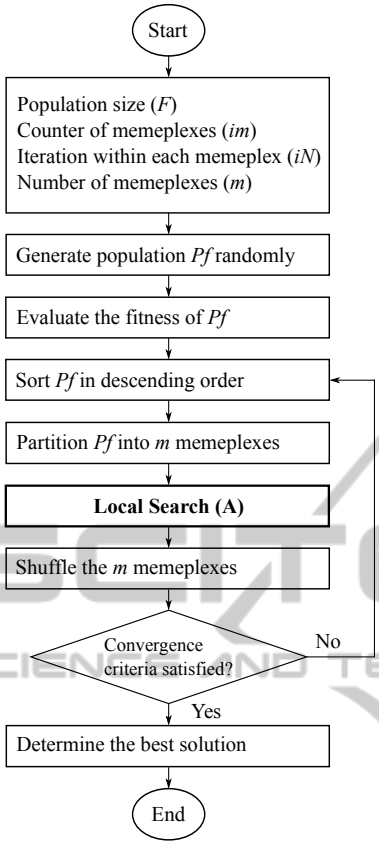


Figure 1: SFLA steps (Illustration adapted from (Amiri et al., 2009)).

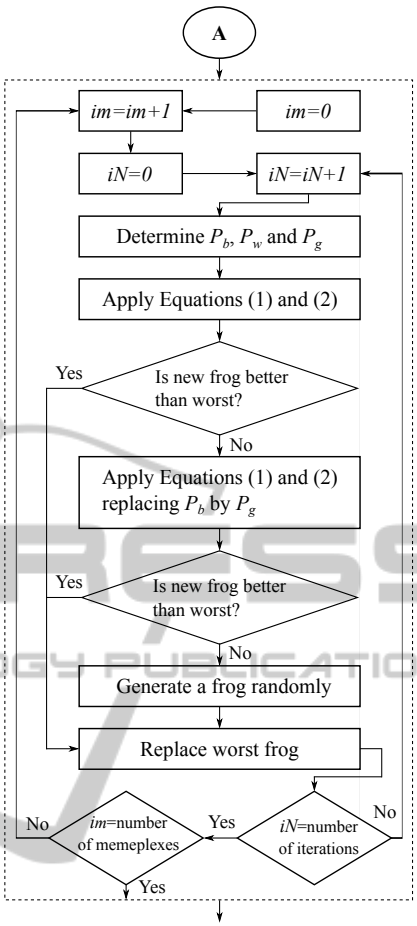


Figure 2: Frog-Leaping local search. (Illustration adapted from (Amiri et al., 2009).)

when searching for feasible periods where to schedule the exams.

3.3 Construction of Submemeplexes

As mentioned previously, the individual frogs in the memeplex are selected to form a submemeplex according to their fitness (Eusuff et al., 2006). The selection strategy is to give higher weights to frogs that have higher performance values and less weight to those with lower performance values. The weights are assigned following a triangular probability distribution:

$$P_j = \frac{2(n+1-j)}{n(n+1)}, \quad j = 1,2,\ldots,n \qquad (3)$$

where n is the number of frogs in the memeplex. As frogs within the memeplex have been previously sorted in descending order of the fitness value, the frog with the best performance has the highest probability $p_1 = 2/(n+1)$ of being selected for the submemeplex, and the frog with the worst performance has the lowest probability $p_n = 2/n(n+1)$. As mentioned before, q distinct frogs are selected randomly from n frogs in each memeplex to form the submemeplex.

3.4 Update the Worst Performance Frog

Inside each submemeplex, the worst performance frog is updated according to the procedure depicted on Figure 3. Part of this procedure is an adaptation of the crossover operator of (Abdullah et al., 2010). As can be seen from figure, this operator produces feasible solutions, so no special constraint-handling techniques such as repairing schemes or penalty function constraint handling are needed.

Executing the steps of the SFLA (Figure 2), the new frog is going to replace the worst frog if it is better than this last one. Otherwise, the procedure of Figure 3 is repeated but substituting P_b by the global best frog, P_g. If the new frog doesn't still improve over the worst frog $U(q)$, then a random solution is generated as the new $U(q)$, replacing the worst frog.

t_1	e_2	e_{14}	e_{10}	e_3	e_{16}
t_2	e_1	e_{11}	e_4		
t_3	e_9	e_{20}	e_5	e_{18}	
t_4	e_6	e_{13}	e_7		
t_5	e_8	e_{12}	e_{15}	e_{17}	e_{19}

(a) Solution $P_b^{'}$

t_1	e_{15}	e_{20}			
t_2	e_9	e_2	e_{12}	e_{10}	e_7
t_3	e_6	e_1	e_{17}	e_{13}	
t_4	e_5	e_{18}	e_4	e_{16}	
t_5	e_8	e_{14}	e_{11}	e_3	e_{19}

(b) Solution P_w

t_1	e_2	~~e_{14}~~	e_{10}	~~e_5~~	e_{16}			
t_2	e_1	e_{11}	e_4	~~e_8~~	e_{14}	~~e_{11}~~	e_3	e_{19}
t_3	e_9	e_{20}	e_5	e_{18}				
t_4	e_6	e_{13}	e_7					
t_5	e_8	e_{12}	e_{15}	e_{17}	~~e_{19}~~			

(c) New solution $P_w^{'}$

Figure 3: Worst frog improvement procedure. (a) First, we copy the best frog, P_b, and eventually mutate it, producing $P_b^{'}$. The new frog is generated by inserting in $P_b^{'}$, at a random period (shown shaded), exams chosen from a random period from solution P_w (as in (b)), producing the new frog $P_w^{'}$ (c). When inserting exams, some could be infeasible or already existing in that time slot (respectively, the case of e_8 and e_{11} in (c)). These exams are not inserted. The duplicated exams in the other time slots are removed. Then this new solution $P_w^{'}$ is further mutated (according to some probability).

Some final details about the procedure illustrated in Figure 3 are now explained. The frog $P_b^{'}$ is a copy of the frog P_b with the following mutation. Two periods of $P_b^{'}$ are selected randomly and the exams are swapped between them. The mutation probability used is $mp_1 = 0.1$. The *number of periods* from P_w where exams are selected for insertion into $P_b^{'}$ is not *one* (as depicted in the procedure, for sake of simplicity), but *three*. Finally, the new solution $P_w^{'}$ is also mutated with the same mutation operator described, but the mutation probability is set to $mp_2 = 0.05$.

3.5 Generation of the Random Frog

To generate the random frog, needed when the new frog is not better than the worst frog, we use the construction method (based on the graph colouring heuristic), described in Subsection 3.2.

4 EXPERIMENTAL RESULTS

In this section we present simulation results of the SFLA on the Toronto benchmark instances. The algorithm was implemented in the C++ language. The parameters of SFLA are: Population size $F = 6000$, Memeplex count $m = 20$, Memeplex size $n = 300$, Submemeplex size $q = n/10 = 30$, Number of time loops $L = 15$. Hardware and software specifications are: Intel Core i7-2630QM, CPU @ 2.00 GHz $\times$ 8, with 8 GB RAM; OS: Ubuntu 12.04, 32 bit; Compiler used: GCC v. 4.6.3.

In the experiment carried out we've set the memeplex size, n, to be much greater than the submemeplex size, q, in an effort for the algorithm be able to escape from local optima. The parameter values were chosen empirically, in a way to achieve a reasonable balance between global and local exploration (Eusuff et al., 2006). The number of time loops L is the number of generations or the number of iterations used as the stopping criteria in the algorithm (Figure 1).

Table 1 present a comparison of the SFLA and other published algorithms. The best results are presented in bold. The SFLA is capable to find feasible timetables with penalty costs comparable to the costs of solutions produced by state-of-the-art algorithms. Figure 4 show the SFLA evolution for three Toronto instances, namely, rye92, sta83, and yor83. The penalty cost and time required to optimize in the tested computer are also shown. With respect to time limits, the fastest instance to be solved was hec92, which took 56 seconds, and the slowest one was pur93, which took 2784 seconds ($\approx$ 46 minutes). Demeester et al. (Demeester et al., 2012), which achieved the best results on several benchmark, have set execution of the algorithm between 1 hour and 12 hours, for the easier and difficult instances, respectively. It is not possible to compare the algorithms because the hardware used is different, but as SFLA obtained the presented results in a very short time, there's still considerable processing time that can be used for application of a more complex and efficient approach.

Table 1: Selection of the best results from the literature compared with the best obtained values from the SFLA approach. The algorithms's authors are: C96 (Carter et al., 1996), M03 (Merlot et al., 2003), B04 (Burke and Newall, 2004), Y05 (Yang and Petrovic, 2005), B08 (Burke and Bykov, 2008), B10 (Burke et al., 2010), D12 (Demeester et al., 2012).

Instance	**SFLA**	C96	M03	B04	Y05	B08	B10	D12
car91	6.04	7.10	5.10	5.00	**4.50**	4.58	4.90	4.52
car92	5.08	6.20	4.30	4.30	3.93	3.81	4.10	**3.78**
ear83	37.31	36.40	35.10	36.20	33.71	32.65	33.20	**32.49**
hec92	11.38	10.80	10.60	11.60	10.83	10.06	10.30	**10.03**
kfu93	16.57	14.00	13.50	15.00	13.82	**12.81**	13.20	12.90
lse91	13.60	10.50	10.50	11.00	10.35	**9.86**	10.40	10.04
pur93	6.81	**3.90**	–	–	–	4.32	–	5.67
rye92	10.96	**7.30**	8.40	–	8.53	7.93	–	8.05
sta83	157.66	161.50	157.30	161.90	158.35	157.03	**156.90**	157.03
tre92	9.21	9.60	8.40	8.40	7.92	7.72	8.30	**7.69**
uta92	4.00	3.50	3.50	3.40	3.14	3.16	3.30	**3.13**
ute92	27.12	25.80	25.10	27.40	25.39	27.79	24.90	**24.77**
yor83	38.52	41.70	37.40	40.80	36.35	34.78	36.30	**34.64**

5 CONCLUSIONS

In the research undertaken we investigated the application of the Shuffled Frog-Leaping Algorithm to the examination timetabling problem. The SFLA is a memetic metaheuristic with global and local search capabilities, and providing a lower number of evaluations of the fitness function compared to genetic algorithms. The key issues in the ETTP are the feasible exploration of the search space and minimisation of a costly fitness evaluation.

The simple worst frog improvement method implemented was able to search the solution space and without disrupting heavily the new frogs found.

The preliminary results obtained in the Toronto benchmark data indicate that the proposed approach give results that are comparable to the ones obtained by state-of-the-art algorithms. On some instances, the algorithm converges prematurely and cannot progress easily (e.g. yor83). In other instances (e.g. rye92 and pur93), due to the characteristics of the dataset, for the tested parameters, the population does not saturate and the algorithm could improve the results further, if more time was given. Another aspect to mention is the time taken by the algorithm to solve the Toronto instances. This time was between approximately 1 minute and 46 minutes for the easiest and difficult instances, respectively. The times reported in literature vary between 1 hour and 12 hours, respectively. Although is not possible to compare algorithms because the hardware is different, we think there's a considerable time gap that could be used in order to apply a more complex approach capable of generating better results.

Our future work will be aimed to improve the algorithm further by incorporating a mechanism for maintaining the memeplex diversity, avoiding the population saturation. Also, we intend to study and implement an efficient neighbourhood or set of neighbourhoods to be applied when creating the new solutions. Finally, we intend to test the algorithm on the International Timetabling Competition datasets (ITC2007).

REFERENCES

Abdullah, S., Turabieh, H., McCollum, B., and McMullan, P. (2010). A tabu-based memetic approach for examination timetabling problems. In Yu, J., Greco, S., Lingras, P., Wang, G., and Skowron, A., editors, *RSKT*, volume 6401 of *Lecture Notes in Computer Science*, pages 574–581. Springer.

Amiri, B., Fathian, M., and Maroosi, A. (2009). Application of shuffled frog-leaping algorithm on clustering. *The International Journal of Advanced Manufacturing Technology*, 45(1-2):199–209.

Burke, E., Eckersley, A., McCollum, B., Petrovic, S., and Qu, R. (2010). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, 206(1):46 – 53.

Burke, E. and Newall, J. (2004). Solving examination timetabling problems through adaption of heuristic orderings. *Annals of Operations Research*, 129(1-4):107–134.

Burke, E. K. and Bykov, Y. (2008). A late acceptance strategy in Hill-Climbing for exam timetabling problems. In *PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*.

Carter, M. and Laporte, G. (1996). Recent Developments in Practical Examination Timetabling. In Burke, E.

and Ross, P., editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 1–21. Springer Berlin / Heidelberg.

Carter, M., Laporte, G., and Lee, S. Y. (1996). Examination Timetabling: Algorithmic Strategies and Applications. *Journal of the Operational Research Society*, 47(3):373–383.

Carter, M. W. (1986). A survey of practical applications of examination timetabling algorithms. *Oper. Res.*, 34(2):193–202.

Demeester, P., Bilgin, B., Causmaecker, P. D., and Berghe, G. V. (2012). A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice. *J. Scheduling*, 15(1):83–103.

Eusuff, M., Lansey, K., and Pasha, F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering Optimization*, 38(2):129–154.

Eusuff, M. M. and Lansey, K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management*, 129(3):210–225.

Merlot, L., Boland, N., Hughes, B., and Stuckey, P. (2003). A hybrid algorithm for the examination timetabling problem. In Burke, E. and Causmaecker, P., editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pages 207–231. Springer Berlin Heidelberg.

Petrovic, S. and Burke, E. (2004). University timetabling. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis, chapter 45*. Chapman Hall/CRC Press.

Qu, R., Burke, E., McCollum, B., Merlot, L. T. G., and Lee, S. Y. (2009). A Survey of Search Methodologies and Automated System Development for Examination Timetabling. *Journal of Scheduling*, 12:55–89.

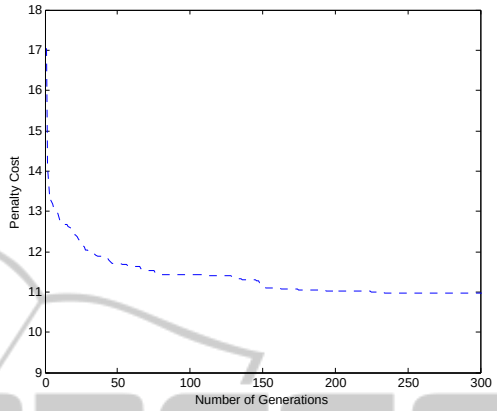
Rahimi-Vahed, A. and Mirzaei, A. H. (2007). A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem. *Computers & Industrial Engineering*, 53(4):642 – 666.

Wang, Y.-m., Pan, Q.-k., and Ji, J.-z. (2009). Discrete shuffled frog leaping algorithm for examination timetabling problem. *Computer Engineering and Applications*, 45(36):40.

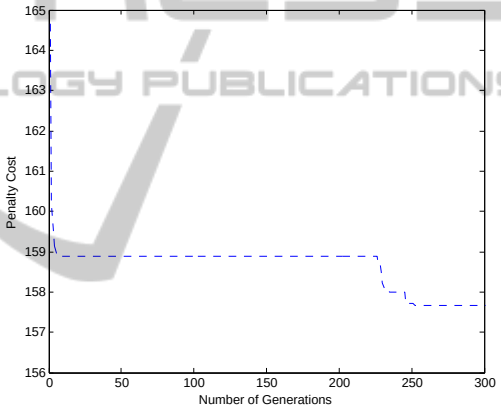
Xu, Y., Wang, L., Liu, M., and Wang, S.-y. (2013). An effective shuffled frog-leaping algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *The International Journal of Advanced Manufacturing Technology*, pages 1–9.

Xue-hui, L., Ye, Y., and Xia, L. (2008). Solving tsp with shuffled frog-leaping algorithm. In *Intelligent Systems Design and Applications, 2008. ISDA '08. Eighth International Conference on*, volume 3, pages 228–232.

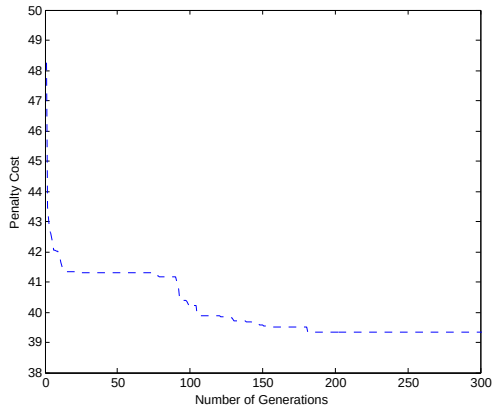
Yang, Y. and Petrovic, S. (2005). A novel similarity measure for heuristic selection in examination timetabling. In Burke, E. and Trick, M., editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 247–269. Springer Berlin Heidelberg.



(a)



(b)



(c)

Figure 4: Convergence of the SFLA on the following Toronto datasets (exact penalty cost achieved and total time taken are also shown): (a) rye92 dataset (took 393 seconds, obtaining a penalty cost of 10.96); (b) sta83 dataset (took 66 seconds, obtaining a penalty cost of 157.66); (c) yor83 dataset (took 189 seconds, obtaining a penalty cost of 39.33).