Adaptive Evolutionary Algorithms and Extensions to the HyFlex Hyper-heuristic Framework

Gabriela Ochoa[1], James Walker[2], Matthew Hyde[2], and Tim Curtois[2]

[1] Department of Computing Science and Mathematics, University of Stirling, UK
[2] School of Computer Science, University of Nottingham, UK

Abstract. HyFlex is a recently proposed software framework for implementing hyper-heuristics and domain-independent heuristic optimisation algorithms [13]. Although it was originally designed to implement hyper-heuristics, it provides a population and a set of move operators of different types. This enable the implementation of adaptive versions of other heuristics such as evolutionary algorithms and iterated local search. The contributions of this article are twofold. First, a number of extensions to the HyFlex framework are proposed and implemented that enable the design of more effective adaptive heuristics. Second, it is demonstrated that adaptive evolutionary algorithms can be implemented within the framework, and that the use of crossover and a diversity metric produced improved results, including a new best-known solution, on the studied vehicle routing problem.

1 Introduction

A hyper-heuristic is *a search method or learning mechanism for selecting or generating heuristics to solve computational search problems* [6]. The main motivation is to develop automated search methodologies with higher generalisation abilities, which will potentially increase their application in practice. The HyFlex (Hyper-heuristic Flexible) framework [13] has been recently proposed to assist researchers in hyper-heuristics and autonomous search control. HyFlex consists of two parts. First, a Java programming interface for hyper-heuristics, which splits the heuristic search process into two modules. One module contains the problem-specific algorithm components and other contains the problem-independent components. Second, a library of *ready-to-use* problem domain modules covering hard combinatorial optimisation problems with a rich variety of search operators and including real-world industrial data. Two important antecedents of the HyFlex framework are the domain-barrier hyper-heuristic conceptual framework [8], and the PISA software framework [3].

Currently, six problem domain modules are implemented in HyFlex (which can be downloaded from the CHeSC 2011 website [1]). These are the original four test domains: permutation flow shop, one-dimensional bin packing, maximum satisfiability and personnel scheduling; and the two additional domains used for

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 418–427, 2012.

the competition: traveling salesman and vehicle routing. HyFlex was used to support an international research competition: the first Cross-Domain Heuristic Search Challenge [1] that received significant international attention.

HyFlex was initially designed to support research within hyper-heuristics. However, since the framework provides search operators of different types (mutation, crossover, ruin-recreate and hill-climbing) approaches not traditionally identified as hyper-heuristics can be implemented using the framework. For example, several adaptive implementations of iterated local search (ILS) within HyFlex have been published recently [4,5,17,7]. Indeed, the algorithms ranking 2^{nd} and 3^{rd} in the 2011 competition can be seen as adaptive ILS methods. These approaches can be considered as hyper-heuristics as they operate in a domain independent fashion, using limited information from the search process and following a modular design. Moreover, they coordinate the effort of several move operators and local search heuristics.

The contributions of this paper are twofold. First, we describe number of extensions to the HyFlex framework that will enable the implementation of more robust and effective adaptive search heuristics. Second, we extend a previous adaptive ILS hyper-heuristic [17], which is a single-point search approach, by incorporating a population and the use of crossover heuristics. This brings hyper-heuristics close to adaptive memetic algorithms [14]. These two approaches have developed independently, but they share several features. In particular, they need to provide adaptive mechanisms to autonomously guide the choice of operators (or memes) during the search. These mechanisms have been also studied within the evolutionary computation community using the term *adaptive operator selection* [9,12].

The next section overviews the proposed extensions to the HyFlex interface, while section 3 describes their implementation within a selected problem domain: vehicle routing. Section 4 describes an empirical study illustrating that: (i) adaptive memetic algorithms can be successfully implemented within the HyFlex framework, and (ii) the distance metric incorporated in HyFlex can be used to implement state-of-the-art adaptive operator selection mechanisms. Finally, section 5 summarises our main findings and discusses routes for future research.

2 Extensions to the HyFlex Interface

Providing additional feedback information from the search process would improve the robustness and effectiveness of adaptive search heuristics. Below we discuss the proposed extensions to the HyFlex interface, including their motivation and an indication of which types of approaches may benefit from these extensions.

Distance between Solutions: An important source of feedback for population-based algorithms is an indication of the genotypic diversity in the population. Moreover, recently proposed adaptive operator selection mechanisms rely on the

population diversity as a source of feedback [12]. In order to calculate the diversity of a population, a distance metric between solutions is needed. Therefore, the HyFlex interface is extended with the following two methods:

```
double getMaxDistance()
double solutionDistance(int solutionIndex1, int solutionIndex2)
```

We assume that the minimum distance between two solutions is zero, and that this occurs when they are exactly the same. Since different representations require different distance metrics and measurement ranges, the method `getMaxDistance` returns the maximum possible distance max_d between two solutions. The method `solutionDistance` returns a value between 0 and max_d representing the distance between the two solutions in the memory of solutions as indicated by the input indices.

Solution Metrics and Alternative Objective Functions: Heuristic search approaches that dynamically modify the fitness function in order to escape local optima or fitness plateaus can be found in the metaheuristics and artificial intelligence literature [2]. Moreover, a recently published hyper-heuristic approach [18], declared the winner of a computational search competition to solve the Eternity II Puzzle, employs alternative fitness functions in order to guide the search. Guided local search (GLS) proposes augmenting the objective function with a set of penalty terms on a set of *solution features* [16]. A solution feature is a non-trivial property of the solution and a cost is associated to each feature. We borrow and extend this concept in HyFlex, instead of feature, we use the term *metric* to refer to additional costs or objectives associated to a given solution. The two following two methods are included:

```
int getNumberOfMetrics()
double getMetric(int solutionIndex, int metricIndex)
```

Where the first method returns the number of solution metrics, and the second gets the value of the given metric for the indicated solution in memory. These metrics can then be used by the hyper-heuristic designer to implement their own alternative objective functions to guide the search.

Additional Instances: The version of HyFlex used in the 2011 competition contains 12 instances for the test domains (the 10 training instances and 2 additional hidden instances), and 10 instance for the new (hidden) domains. Moreover, this instance data is included within the software, and there is no flexibility for adding new instances. Having additional instances will both improve the development of robust online strategies, and facilitate the implementation of offline configuration techniques. An approach based on offline learning for algorithm selection obtained surprisingly good results in the 2011 challenge [11]. This is very promising, as the challenge was designed to encourage online approaches to heuristic selection. To enable the incorporation of additional instances the method: `void loadInstanceFromFile (String fileName)` is included in HyFlex, which loads the instance indicated in the file and set is as the current instance. The file needs to have the correct format, which will be included in the domain documentation.

Additional Utilities: Utilities for saving and retrieving solutions from files may facilitate both the reuse of previously found solutions and the analysis of previous runs. The two methods below are included:

```
void loadSolutionFromFile(String fileName, int solutionIndex)
void SolutionToFile(String fileName, int solutionIndex)
```

Where `solutionIndex` refers to the position in the memory of solutions, and `fileName` to the name of the source or destination file.

3 The Extended Vehicle Routing Domain

The vehicle routing problem with time windows involves satisfying the demand of a set of customers, using the fewest possible vehicles, and adhering to all constraints such as time windows, whereby a customer must be served between two points in time. Each vehicle starts from the same point, the depot. A route consist of a list of locations. The HyFlex VRP problem domain [17] provides 12 search operators including: 4 mutation, 2 ruin-recreate, 4 hill-climber and 2 crossover heuristics. The objective function balances the dual objectives of minimising the number of vehicles, and minimising the total distance travelled. Due to space constraints we refer the reader to [17] for a complete description. We concentrate here on the problem domain extensions.

Distance Metric: We implemented a distance metric suggested in [10], which is based on a concept formulated for the travelling salesman problem. The metric considers the number of common edges between two solutions. For the vehicle routing problem, an edge represents an undirected link between two locations. The distance metric produces a value between 0 and 1 and the formula is as follows: $distance = \frac{totalEdges - commonEdges}{totalEdges}$.

Solution Features: The solution features provided are: (1) the default objective function, which is a weighed sum of the number of routes and the distance traveled,(2) the number of routes or vehicles, (3) the total distance traveled, and (4) the distance of the shortest route.

Instance File Format: The instance format is the Solomon format. The instance file provides the number of customers and vehicle capacity. This is followed by a list of customers with he following attributes: (1) customer number, (2) X co-ordinate, (3) Y co-ordinate, (4) demand, (5)ready time, (6) due date, (7) service time.

4 Empirical Study

4.1 Algorithms

Two classes of algorithms are considered: adaptive iterated local search and adaptive memetic algorithms. These algorithms adapt the probabilities associated to the available search operators, according to the history of their performances. The operators are then selected according to these learned probabilities

using a roulette wheel mechanism. Since HyFlex provides several operators belonging to different classes: mutation, ruin-recreate, crossover and hill-climbing; several adaptive mechanisms may be required for selecting different operators at different parts of an algorithm framework. This study consider two variants of each algorithm class, which differ on the feedback information used from the search process to adapt the choice of search operators. The first variant considers only the fitness function improvements or deteriorations obtained after applying the search operators, while the second is based on the *compass* mechanism [12], which considers a diversity metric and the running time of the operators in conjunction with their fitness variation as sources of feedback. The underlying idea behind the compass control mechanism is to provide an adequate exploration/exploitation balance. Thus, both diversity and quality are pertinent criteria to guide the search.

Adaptive Iterated Local Search: We consider the best performing algorithm proposed in [17], which is a multiple neighborhood ILS algorithm that includes adaptive mechanisms for both the perturbation and improvement stages. The perturbation stage selects among the set of available mutation and ruin-recreate heuristics using the *extreme value* [9] operator selection mechanism. The improvement stage considers the available hill-climbers and incorporates a simple adaptive mechanism, in which the operators are ordered according to learned propoabilities and sequentially applied using this order. We name this algorithm *AILS*. A new version of this algorithm is implemented, in which the extreme value mechanism is substituted by the by the compass mechanisms. We call this algorithm *AILS-C*.

Adaptive Memetic Algorithm: Our implementation of adaptive memetic algorithms works as follows (see Algorithm 1). A small population (of size 4) is generated and then goes through a recombination stage in which all possible recombination pairs are considered and a randomly selected crossover operator (from the available pool) is applied for each pair. From all these generated solutions the best four are kept. This is a costly stage and it is only invoked a number of times during the search process. A perturbation and improvement stage follows. For each member of the population, a mutation or ruin-recreate heuristic is selected from the pool according to operator probabilities learned using a simple reinforcement learning mechanism. The solution is thereafter improved by a hill-climbing heuristic. The improvement heuristic to apply is also selected according to learned probabilities. We call this algorithm *AMA*. A variant is also implemented in which the reinforcement learning mechanism used in the perturbation stage is substituted by the compass mechanism. We call this algorithm *AMA-C*.

4.2 Results

The experiments were conducted using the 10 VRP test instances currently available in the 2011 HyFlex software. These instances were originally taken

Algorithm 1. *Adaptive Memetic Algorithm (AMA).*

```
P = GenerateInitialPopulation
repeat
  P'= RecombinationStage(P)
  P'' = MutationAndImprovementStage(P')
  UpdatePerturbationOperatorProb
  UpdateImprovementOperatorProb
  P = SelectBest(P' + P''')
until time limit is reached
```

from [15] and include 5 instances from the Solomon data set and 5 from the Gehring and Homberger data set. Both data sets include three types of instances: **R**andom, **C**lustered, and **R**andom **C**lustered; according to the way in which the customers' locations are determined. Details about the instances can be found in the first three columns of Table 1. In the instance name, the first number indicates the HyFlex numbering, while the first letter whether it is a Solomom (S) or Homberger (H) instance. The second group of letters indicates the type of instance; and the final string corresponds to the identifier in the data set.

As a first test, we compared our two base adaptive algorithms AMA and $AILS$ against the best-performing algorithms for VRP in the 2011 competition and using the original HyFlex version. We considered the competition experimental setting, namely, 10 minutes per run, 31 runs per instance and the 5 competition instances. These are instances 1, 2, 5, 6 and 9. Since the instances have different objective function ranges, we selected ordinal data analysis to compare the algorithms. If m is the number of instances and n the number of competing algorithms. For each instance an ordinal value o_k is given representing the rank of the algorithm ($1 \leq o_k \leq n$). An algorithm having a rank o_k in a given instance is simply given o_k points, and the total score of an algorithm is the sum of its ranks o_k across the m instances (this metric is known as the *Borda* count). In this comparison, the number of instances $m = 5$ and the number of algorithms $n = 5$. Therefore, best possible score is 5, and the worst possible is 25. The ranks were calculated according to the median best objective function value across the 31 runs per instance. Figure 1 (a) illustrates the Borda counts for AMA, $AILS$ and the top 3 performing competitors in the 2011 challenge. Clearly, the AMA is the best performing algorithm, producing an almost perfect score.

The next set of experiments use the the new HyFlex VRP domain and the four algorithm variants described above, *AILS*, *AILS-C*, *AMA* and *AMA-C*. The whole set of 10 instances were used (see Table 1). The running time was set to 20 CPU minutes and 10 runs were conducted per instance. The machine running the tests has a 2.27 GHz Intel(R) Core(TM) i3 CPU and 4GB RAM. The Borda count is used for comparison and the median best objective function value is used for the ranking. This time we have $m = 10$ instances and $n = 4$ algorithms. Therefore, the best possible score is 10 and the worst is 40. Figure 1 illustrates the results. We can see that the two versions of the adaptive memetic algorithm have similar performance, and clearly outperform the adaptive ILS algorithms.

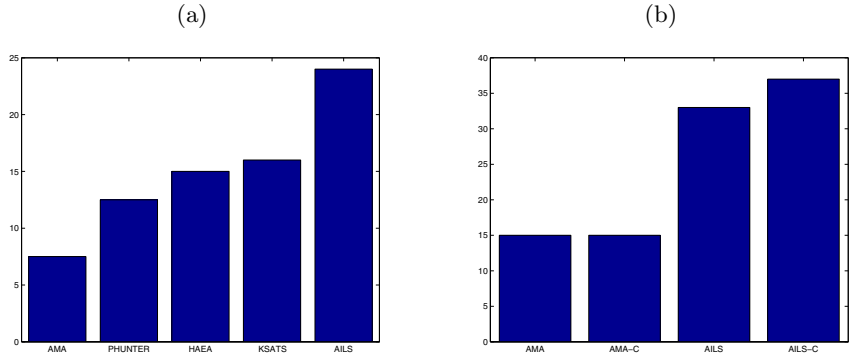

Fig. 1. Borda counts for (a) *AMA*, *AILS* and the top 3 VRP hyper-heuristics in the 2011 challenge, (b) the two variants of *AMA* and *AILS* on the full set of 10 instances and using the extended HyFlex VRP domain. Objective is minimisation.

The boxplots shown in Figure 2 illustrate the magnitude and distribution of the best objective values for 2 representative Homberger instances (instances 6 and 9). Each plot summarises the result of 10 runs from each algorithm. For both instances, the *AMA* algorithms produce the best results. The difference in performance is more noticeable for instance 6, but this behaviour is consistent across all the instances. The Borda counts in Figure 1, indicate that the two versions of *AMA* have similar performance considering the median best objective value. However, the best solutions were in most cases obtained by the $AMA-C$ variant as can be seen in Figure 2 and Table 1.

Finally, Table 1 shows the best solutions found by our *AMA* algorithms together with the best-known solutions for the these instances. The adaptive memetic algorithms matched the bet-known number of vehicles for all the Solomon instances and for two of the Homberger instances. Moreover, for

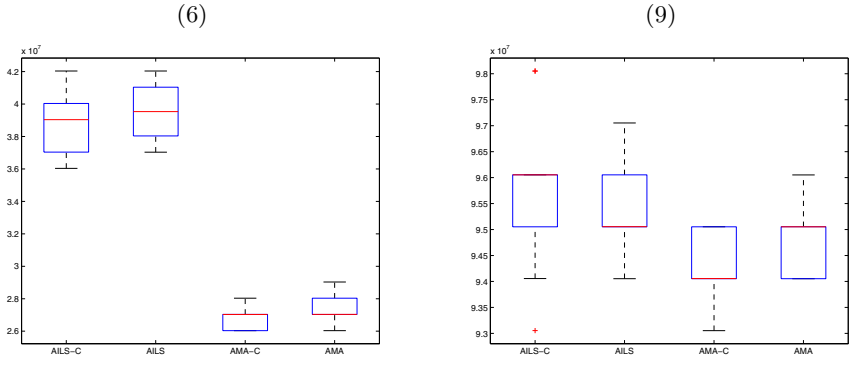

Fig. 2. Distribution of objective function values for Homberger instances 6 and 9. Objective is minimisation.

Table 1. VRP Instances. *AMA* best results vs. best-known results.

Instance			No. of Vehicles			Distance		
Name	Cust.	Capacity	*AMA*	*AMA-C*	*Best-k*	*AMA*	*AMA-C*	*Best-k*
0-SRC207	100	1000	4	**3**	**3**	**1047.42**	1133.83	1061.14
1-SR101	100	1000	**19**	**19**	**19**	1650.8	**1631.82**	1645.79
2-SRC103	100	200	**11**	**11**	**11**	1276.82	1263.78	**1261.67**
3-SR201	100	200	**4**	**4**	**4**	1261.043	1276.45	**1252.37**
4-R106	100	1000	**12**	**12**	**12**	1268.93	1284.23	**1251.98**
5-HC1-10-1	100	200	**100**	**100**	**100**	42481.26	42485.04	**42478.95**
6-HRC2-10-1	250	1000	26	26	**20**	33272.57	**32839.49**	63373.15
7-HR1-10-1	250	200	**100**	**100**	**100**	59020.74	60517.21	**53904.23**
8-HC1-10-8	250	200	101	101	**93**	44037.96	44120.54	**42499.59**
9-HRC1-10-5	250	200	94	93	**90**	52581.52	52439.09	**46631.89**

instance 1 (1-SR101) $AMA-C$ produced a shorter distance, with the same number of vehicles, which makes this a new best-known solution for this instance. Better distances were found for instances 0 and 6, but at the expense of a larger number of vehicles. These results are encouraging as HyFlex was designed to explore adaptive search heuristics that operate in a domain-independent way.

5 Conclusions

We have presented a number of extensions to the HyFlex framework that will enable the implementation of more effective adaptive heuristics, while maintaining a high degree of modularity between the problem-independent and the problem-dependent heuristic components. In particular, the new version supports the implementation of: (i) population-based approaches and mechanisms for operator selection that consider diversity metrics in the solution space, (ii) adaptive approaches that modify the fitness function or consider alternative objective functions, and (iii) offline approaches and portfolio methods that benefit from a greater number of problem instances. This article concentrated on the first of these extensions, namely using a diversity metric to implement more sophisticated adaptive operator selection mechanisms. Our results suggest that this mechanism may improve the search, in particular for locating best solutions. Indeed a new best-known solution was found for one of the studied instances. In future work we plan to further exploit this and the additional HyFlex features. Our HyFlex adaptive evolutionary algorithms also supports that using a population and crossover operators may improve the search. This is an important result, which may encourage the evolutionary computation community, as iterative hyper-heuristics have been traditionally single-point approaches.

The proposed HyFlex extensions were implemented and tested in a single domain: the vehicle routing problem. Work is in progress to incorporate these extensions in other domains such as permutation flow-shop, 1D bin packing and personnel scheduling. We envisage the incorporation of new challenging and real-world domains in HyFlex. We are also planning a second international challenge with additional features. The creativity and enthusiasm of the 2011 competitors pushed

the boundary of hyper-heuristic research. We expect that the new competition will bring the interest and participation not only of hyper-heuristic researchers, but also researchers in reactive search, intelligent optimisation, adaptive operator selection, adaptive memetic algorithms, co-evolutionary memetic algorithms, guided local search, adaptive large neighborhood search, autonomous search, self-* search and automatic configuration of search heuristics to name a few.

Acknowledgements. We would like to thank the HyFlex users and CHeSC competitors for their feedback and suggestions. Our special thanks to J. Kubalic, A. Lehrbaum, K. McClymont, D. Meignan, M. Misir, A. Nunez, E. Ozcan, A. J. Parkes, K. Sim, T. Urli, T. Wauters and F. Xue.

References

1. The Cross-domain Heuristic Search Challenge (CHeSC 2011). Website (2011), http://www.asap.cs.nott.ac.uk/external/chesc2011/
2. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization. Operations research/Computer Science Interfaces, vol. 45. Springer (2008)
3. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA – A Platform and Programming Language Independent Interface for Search Algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
4. Burke, E.K., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vazquez-Rodriguez, J.A., Gendreau, M.: Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms. In: IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, pp. 3073–3080 (July 2010)
5. Burke, E.K., Gendreau, M., Ochoa, G., Walker, J.D.: Adaptive iterated local search for cross-domain optimisation. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011, pp. 1987–1994. ACM, New York (2011)
6. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.: A Classification of Hyper-heuristic Approaches. In: Handbook of Metaheuristics, vol. 146, ch. 15, pp. 449–468. Springer (2010)
7. Chan, C.Y., Xue, F., Ip, W.H., Cheung, C.F.: A hyper-heuristic inspired by pearl hunting. In: International Conference on Learning and Intelligent Optimization (LION 6). LNCS, Springer (to appear, 2012)
8. Cowling, P.I., Kendall, G., Soubeiga, E.: A Hyperheuristic Approach to Scheduling a Sales Summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)
9. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Extreme Value Based Adaptive Operator Selection. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN X. LNCS, vol. 5199, pp. 175–184. Springer, Heidelberg (2008)
10. Kubiak, M.: Distance measures and fitness-distance analysis for the capacitated vehicle routing problem. In: Metaheuristics. Operations Research/Computer Science Interfaces Series, vol. 39, pp. 345–364. Springer US (2007)
11. Mascia, F., Stutzle, T.: A non-adaptive stochastic local search algorithm for the chesc 2011 competition. In: Proceedings of Learning and Intelligent Optimization 6th International Conference, LION 6, Paris, France, January 16-20. LNCS, Springer (to appear, 2012)

12. Maturana, J., Saubion, F.: A Compass to Guide Genetic Algorithms. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN X. LNCS, vol. 5199, pp. 256–265. Springer, Heidelberg (2008)
13. Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J.A., Walker, J., Gendreau, M., Kendall, G., McCollum, B., Parkes, A.J., Petrovic, S., Burke, E.K.: HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search. In: Hao, J.-K., Middendorf, M. (eds.) EvoCOP 2012. LNCS, vol. 7245, pp. 136–147. Springer, Heidelberg (2012)
14. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: a comparative study. IEEE Transactions on Systems, Man, and Cybernetics, Part B 36(1), 141–152 (2006)
15. SINTEF. VRPTW benchmark problems, on the SINTEF transport optimisation portal. Website (2011), `http://www.sintef.no/Projectweb/TOP/Problems/VRPTW/`
16. Voudouris, C., Tsang, E.: Guided local search and its application to the traveling salesman problem. European Journal of Operational Research 113(2), 469–499 (1999)
17. Walker, J.D., Ochoa, G., Gendreau, M., Burke, E.K.: Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In: International Conference on Learning and Intelligent Optimization (LION 6). LNCS. Springer (to appear, 2012)
18. Wauters, T., Vancroonenburg, W., Vanden-Berghe, G.: A guide-and-observe hyper-heuristic approach to the eternity ii puzzle. Journal of Mathematical Modelling and Algorithms, 1–17 (2012), doi:10.1007/s10852-012-9178-4