An Ant Algorithm Hyper-heuristic

E. K. Burke G. Kendall R. F. J. O'Brien† D. Redrup E. Soubeiga

Automated Scheduling, optimisAtion and Planning Research Group,
School of Computer Science and Information Technology, The University of Nottingham,
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, United Kingdom
{ekb, gxk, rob, exs}@cs.nott.ac.uk
† Corresponding author

Ant algorithms were introduced in 1996 by Dorigo as a collaborative optimisation method, but it can be argued that they have not yet fulfilled their potential. Hyper-heuristic research is concerned with the development of "heuristics to choose heuristics" in an attempt to raise the level of generality at which optimisation systems can operate. In this paper the two are brought together. Variants of applications of the ant algorithm as a hyper-heuristic are investigated and developed. The results are evaluated against other hyper-heuristic methods.

1 Introduction

Ant Algorithm

The ant algorithm is motivated by the way in which colonies of ants lay pheromone trails in order to find the shortest route between two points (as described, and applied to route-planning problems such as the Travelling Salesman Problem, in [5]).

The ant algorithm technique can be explained by considering a graph which represents the landscape to be traversed. In this graph, the edges represent paths and all vertices represent choices in route. The graph is then populated by ants which move from one vertex to the next, laying down a pheromone trail based upon the distance travelled: a short distance produces a high pheromone level, a longer distance produces a lower level. At each decision point the ant considers which new path to take randomly, but a greater probability is attached to paths of higher pheromones. Over time these pheromone trails evaporate, so a path once used but since forgotten will eventually have no traces of pheromone at all.

The inferred goal of this method is that shorter routes will grow in popularity as the pheromone trails grow higher, but that the route must be continually traversed in order to maintain its status as the current best route.

In the case of the Travelling Salesman Problem, the ants' selections of possible immediate

destinations is limited by the number of vertices already visited, as they search for the tour of highest quality. As certain ants will be restricted from following certain common routes, alternative routes will continually be explored until all ants converge onto the best available tour.

Further material on ant algorithms can be found in [1] and [8].

Hyper-heuristic

The term *hyper-heuristic* can be used to specify a heuristic which chooses between heuristics. It is possible to employ terminology which refers to the hyper-heuristic as the *high-level heuristic*, and to the heuristics it chooses between as *low-level heuristics*.

Hyper-heuristic research is motivated by the goal of developing systems that are capable of being applied to a wide range of problems. The aim is to develop systems that are not dependent upon problem domain-specific knowledge. Instead the objective is for hyper-heuristic based systems to "pick" the right heuristic (or sequence of heuristics) for the right situation. A detailed analysis of hyper-heuristics, together with examples of recent hyper-heuristic research, can be seen in [2].

In [4, 7] Cowling, Kendall and Soubeiga went on to create a number of hyper-heuristics, which were grouped into loose categories:

- Random, where the selection of a particular low-level heuristic is essentially random, rather than examining any individual merits the heuristics may have;
- Greedy, where (at various decision points) several low-level heuristics are implemented and the best offered projection is accepted;
- Choice Function, where the low-level heuristics are assessed on the degree of improvement they produce alone and in tandem with other heuristics, and thereby selected with more care.

The research went on to show that a number of Choice Function hyper-heuristics performed significantly better than hyper-heuristics from the Random category and the manual efforts of humans working on the same problem.

In [3] Burke, O'Brien et al. made use of the pheromone element of the ant algorithm in a greedy hyper-heuristic, in order to give preference to low-level heuristics with good performance, dismissing those of poor performance and therefore decreasing the computational workload. Low-level heuristics could be re-instated during a re-introduction phase which took place after all active low-level heuristics ceased to produce improvements in the solution.

In [6] Gutjahr suggested the use of an ant algorithm to create a tour of moves, similar to the TSP, to construct or iteratively improve a solution to a given problem.

The natural progression of this research was to investigate and develop the full ant algorithm as a hyper-heuristic, and the resulting Ant Algorithm Hyper-heuristic is presented in this paper.

The rest of the paper is organised as follows: Section 2 describes the Presentation Scheduling Problem (PSP) used by Cowling, Kendall and Soubeiga [4], in order to compare the hyper-heuristics; Section 3 describes the algorithmic basis of the Ant Algorithm Hyper-heuristic, explores some possible problems with the basic algorithm, and presents several variants of the algorithm to be compared; Section 4 compares the results received from applying these variant ant algorithm hyper-heuristics to the PSP; Section 5 concludes the paper.

2. Presentation Scheduling Problem

This problem is described in depth in [4]; we will just present a brief overview of the problem here.

Final year students on the single honours computer science degree at the University of Nottingham students undertake a project which is supervised by an academic member of staff. They are required to give a 15-minute presentation on that project as part of their assessment. Each presentation is marked by three members of staff, known as the First Marker (or Chair), Second Marker, and Observer. Ideally in each presentation either the Chair or the Observer should be the student's supervisor, but often this will not be the case in practice. The format of these presentations has developed into dividing the students into hourly sessions (i.e. up to four presentations per session), in an available seminar room.

The problem is to determine all (student, 1st marker, 2nd marker, observer, session, room) tuples. To formulate the problem, we denote by **I** the set of students, **S** the set of lecturers, **Q** the set of sessions and **R** the set of seminar rooms. The main decision variables are denoted x_{ijklqr} ($i \in \mathbf{I}$, $j,k,l \in \mathbf{S}$, $q \in \mathbf{Q}$, $r \in \mathbf{R}$) where x_{ijklqr} is 1 if the presentation of student i is assessed by First Marker j, Second Marker k and Observer l, and allocated to session q in seminar room r, otherwise x_{ijklqr} is 0.

The problem constraints are:

- Each presentation must be scheduled once.
- There must be at most four presentations for each session and room.
- No staff member may be scheduled to 2 different rooms within the same session.

The problem formulation is given in [4] as a *minimisation* problem with E, the overall objective function made up of 4 weighted goals:

- Fair distribution of presentations per staff member.
- Fair distribution of sessions per staff member.
- Fair distribution of the number of 'early' (before 10:00am) and 'late' (after 4:00pm) sessions per staff member.
- Matching of staff research interest to project topic, and where possible involvement of supervisors in corresponding presentations.

We use the same low-level heuristics as in [4]. They are simple and based around moving, replacing or swapping an object:

- (*h1*): Replace a random lecturer *j1* with another random lecturer *j2* in a random session *q* during which *j1* is scheduled for presentations.
- (*h2*): Same as *h1*, but *j1* has the largest number of scheduled presentations.
- (*h3*) Same as *h2*, but *q* is the one where *j2* has the smallest number of presentations.
- (*h4*) Move a random presentation *i* from its current session-room into another random session-room *q*-*r*.
- (*h5*) Same as *h4*, but presentation *i* is that for which the sum of presentations involving all three staff members (1st marker, 2nd marker, observer) is smallest of all sessions.
- (*h6*): Same as *h5* but session *q* is one where at least one of the staff members (1st marker, 2nd marker, observer) is already scheduled for presentations.
- (*h7*): Swap 2nd marker of one presentation with observer of another (although supervisors may not be removed).
- (*h8*): Swap 1st marker of one presentation with 2nd presentation of another (again, supervisors may not be removed).

3. Ant Algorithm Hyper-Heuristic

The ant algorithm technique makes use of a graph G, the vertices V in that graph, a set of edges E which connect the vertices together, and a set of ants A which will traverse the edges, evaluate the route, and lay a pheromone trail on those edges to convey that assessment to other ants. Graph G has the properties that it is complete, directed and self-directed, i.e. that for any pair of vertices *i* and *j*, including the case where $j = i$, there exists a directed edge from *i* to *j*.

In the ant algorithm hyper-heuristic we use the ant algorithm as an analogy for the hyper-heuristic. Each ant corresponds to a potential solution, and its journey corresponds to the construction of that solution. The hyper-heuristic will therefore be producing |A| solutions to the problem, and from these will output the best solution-state reached during the run.

In our analogy, every vertex in the graph represents a low-level heuristic. There are

therefore |V| low-level heuristics. As each ant journeys to a new vertex, it implements the low-level heuristic represented by the destination vertex on the solution it (that is, the ant) represents.

The important stage in the hyper-heuristic is that of choosing which heuristic to implement next upon the solution. This corresponds to an ant at a vertex analysing all edges leading away from that vertex to decide which is a good edge to traverse next. Each edge represents various domain-free information which will be available to an ant about its source and destination vertices, such as the CPU time required to set-up and implement the destination heuristic. There will also be a pheromone trail on some edges which represents the assessment of other ants on the degree of improvement gained on a solution by implementing the source heuristic followed immediately by the destination heuristic.

So, an ant k journeys to vertex i, and implements low-level heuristic i upon the solution represented by k. Ant k will then choose a new low-level heuristic j, selecting randomly but with bias towards good edge information (e.g. low CPU requirements) and high pheromone levels (i.e. the confidence level other ants have shown in the belief that implementing low-level heuristic j after low-level heuristic i is a good move).

At the same time the existing pheromone trails will decay. Edges which represent *bad* decisions (i.e. decisions where other ants have not expressed high confidence in the belief that implementing j after i was a good move) should therefore be used less as time passes. Edges which represent good decisions (i.e. decisions where the ants have expressed high confidence) will continue to be used and will maintain that confidence, and the ants will continue to be influenced by that good decision in the future.

After traversing each edge, each ant will implement the low-level heuristic of the destination vertex, evaluate the degree of improvement of its solution, and lay a pheromone trail over the previous edge corresponding to the evaluation, thereby letting other ants who visit the source vertex know how good a decision it believes travelling to this vertex was.

This “simplest” version of the ant algorithm hyper-heuristic shall be known as *AAHHOneEdgeTour* in this paper (the reasons for this will become clearer after more complex variants of the hyper-heuristic are described.

It is easy to see that as the graph develops, certain paths will become more popular than others, indicating that certain sequences of low-level heuristics will produce significant improvements and that the ants are sharing this knowledge between them. This popularity will be tempered by the pheromone decay which occurs when paths are not applicable at other solution-construction states.

There is an inferred trade-off in this hyper-heuristic, since we do not know how many ants will be optimal for any given problem. Factors which may affect this optimality include the number of low-level heuristics being used. A large colony of ants will learn far quicker than a small colony, and produce a large number of solutions, but they will also incur greater computational costs.

We also demonstrate a weakness to laying pheromones after *each* implementation of a low-level heuristic (i.e. after each edge). A low-level heuristic which decreases the quality of the solution, but which provides an escape from local optima, will still be reflected badly within the graph because of its stand-alone effect. This will limit the use of the low-level heuristic until the quality of improvement produced by other low-level heuristics becomes so low that the probabilities become nearly equal.

To offset this we introduce to each ant a new attribute: an integer value which informs the ant of how long its journey will be. For example, an ant with a journey length of 3 will repeat the procedure of "choose low-level heuristic, travel to it, implement it" 3 times before evaluating its journey. Another ant with a journey length of 6 will implement twice as many low-level heuristics before its own self-evaluation. The low-level heuristic's usefulness will then be considered as part of a journey in the hope that a sequence which includes individually *poor* moves may produce a significantly better move when combined. This version of the hyper-heuristic is called *AAHHFixedLengthTour* in this paper (*AAHHOneEdgeTour* can be seen as a special case of *AAHHFixedLengthTour* where every ant always pursues a journey of length 1).

Each ant, and therefore each solution, is now associated with a fixed journey length, which may or may not be beneficial. Solutions whose ants make longer journeys may benefit more from other ants' experiences, but solutions whose ants make shorter journeys may in turn be producing more shallow evaluations, which may hinder the progress of their own solutions and (through the graph) hinder all the other solutions as well. At this stage in our research this hypothesis is untested, but we suggest several other variants which allow an ant to choose a new journey length after each journey. We use J to represent a bag of integer values to be used as journey lengths (we use bags instead of sets because duplicate values are permissible), and we assume the bag is sorted in order from smallest value to largest.

In our third variant, named *AAHHRandomLengthTour*, each ant selects a random new journey length from J upon completion of its previous journey.

In the fourth variant, *AAHHPermutation*, we assume the bag of journey lengths has at least as many elements as there are ants (i.e. $|J| \geq |A|$). Initially each ant receives a distinct journey length from J. When an ant finishes its journey, it swaps its current journey length for a random value in J not currently taken by another ant (in the case that $|J| > |A|$). If $|J| = |A|$ the ant must

pause and wait for another ant to end its journey, at which point the two ants swap. The distribution of journey lengths will therefore always remain a permutation of the bag's initial contents, thereby maintaining an even distribution of journey lengths while allowing ants to follow different journey lengths.

So far in our research, we have based our formulae for choosing between edges and updating pheromone trails directly upon the work of Dorigo [5].

Each ant chooses the next low-level heuristic it will implement on its solution by random selection: we formulate the transition probability p^k_{ij} for ant k travelling from vertex (low-level heuristic) i to vertex (low-level heuristic) j during iteration t as

$$p^k_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{v} [\tau_{iv}(t)]^{\alpha} \cdot [\eta_{iv}]^{\beta}} \quad (1)$$

where $\tau_{ij}(t)$ is the amount of pheromone on edge (i, j) at time t, and η_{ij} is defined as 1 divided by the amount of CPU time to implement low-level heuristic j after low-level heuristic i, and α and β are parameters to control the relative importance of trail vs. edge information.

We formulate the new pheromone level of each edge after a journey evaluation as

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij} \quad (2)$$

where ρ is a coefficient such that $(1 - \rho)$ represents the evaporation of the trail between t and $t+1$,

$$\Delta\tau_{ij} = \sum_{k}^{|A|} \Delta\tau^k_{ij} \quad (3)$$

where $\Delta\tau^k_{ij}$ is the quantity of pheromone laid on edge (i, j) by the k-th ant between time t and $t+1$. It is given by

$$\Delta\tau^k_{ij} = \frac{Q \cdot I \cdot N}{L_k} \quad (4)$$

where Q is a constant (chosen to help maintain pheromone strength against evaporation), I is the total improvement of the k-th ant's solution over its journey, L_k is the length of the k-th's ant's journey and N is the number of times in that journey that the edge was traversed ($0 \leq N \leq L_k$).

We suggest two possible alternatives to this method, since it rewards good and bad

moves equally within a tour, which defeats the purpose of promoting only good routes.

The first is to rank the low-level heuristics along the tour in order of the amount of improvement they perform, on a scale between 1 and the length of the tour. I.e. on a four-heuristic tour, the best-performing low-level heuristic would be given 4 points, the second-best 3, the third-best 2 and the worst 1. This value is then assigned to Q. The reverse is applied if the overall solution is not improved, thereby exercising the maximum penalty on the worst performers.

The second alternative introduces a second type of ant into the colony. This ant is a *greedy ant*: it always selects, at any given decision point, the best low-level heuristic to implement, regardless of the pheromone trails of other ants. While this method increases the computational overload of the method it does have the virtue of producing a solution at least as good as the greedy hyper-heuristic (allowing for the extra computational workload).

4. Experiments

All algorithms reported in this paper were coded in Microsoft Visual C++ version .NET and all experiments were run on a PC Pentium III 1800MHz with 256MB RAM running under Microsoft Windows 2000 version 5.

We describe two sets of experiments. In the first set, we aim to make a direct comparison between the ant algorithm hyper-heuristics presented in this paper with other hyper-heuristics previously established in the literature. In the second set we intend to investigate more closely the low-level behaviour of the ant-algorithm hyper-heuristics. So far we are able to compare the following six hyper-heuristics: with the following hyper-heuristics: *SimpleRandom* (SR) [4], *RandomPermDescent* (RPD) [4], *Greedy* (G) [4], *AAHHOneEdgeTour* (AOET), *AAHHFixedLengthTour* (AFLT) and *AAHHRandomLengthTour* (ARLT). All hyper-heuristics start from the same initial solution, produced from the constructive heuristic (ch) detailed in [4].

For all algorithms we distinguish the cases where all moves (am) are accepted and those where only improving moves (oi) are accepted. Results (averaged over 5 runs) are given in Table 1 for three instances *csit0*, *csit1* and *csit2*, the same data sets used in [7]. All experiments take 10 minutes.

In the ant algorithm hyper-heuristics we set |A| to 10, ρ to 0.9, α to 5 and β to 0 (initially, in deference to the high speed of the low-level heuristics and therefore the significantly high values of η_{ij}). AFLT and ARLT use the bag J of values 1 to 10. Numbers in brackets refer to the various kinds of pheromone-laying functions: in (1) Q is set to 1, in (2) Q is set to the ranking of

the low-level heuristic. For AFLT we also use (3) and (4), which are the same as (1) and (2) respectively except for the addition of the greedy ant, which is always the ant set to a fixed tour-length of 1.

Table 1:

Dataset:		csit0			csit1			csit2	
ch		-557.5			-2596.4			-1470.7	
HH	Worst	Average	Best	Worst	Average	Best	Worst	Average	Best
AOET(1)	-611.1	-699.12	-776.5	-2649.1	-2676.26	2724.4	-1822.6	-1927.38	-1993.2
AOET(2)	-642.5	-710.5	-766.5	-2642.4	-2686.98	2749.7	-1915.8	-2000.88	-2157.9
AFLT(1)	-627.3	-649.82	-685.5	-2652.1	-2680.44	-2705.4	-1725.6	-1969.4	-2107.4
AFLT(2)	-598.5	-685	-726.5	-2661.4	-2688.5	-2727.4	-1975.6	-2059.2	-2172.8
AFLT(3)	-1053	-1099.92	-1127.6	-3113	-3178.4	-3211.5	-3079.9	-3157.32	-3184.8
AFLT(4)	-1069.8	-1111.82	-1129.3	**-3159**	**-3191**	**-3221.5**	**-3138.4**	**-3172.22**	**-3190.3**
ARLT(1)	-642	-682.54	-755	-2653.4	-2680.78	-2696.4	-1752.7	-1967.44	-2099.7
ARLT(2)	-622.7	-689.9	-785	-2639.4	-2672.88	-2712	-1791.8	-1941.1	-2047.8
SR(am)	-725.7	-756.4	-793.4	-2596.4	-2610.14	-2651.3	-1470.4	-1522.6	-1602.1
SR(oi)	**-1112**	-1116.7	-1120.5	-3117.4	-3135.68	-3160.2	-3080.4	-3105.62	-3120.3
RPD(am)	-670.7	-739.6	-787	-2596.4	-2625.92	-2655	-1470.7	-1539.02	-1612.7
RPD(oi)	-1103.5	-1111.4	-1120.5	-3113.3	-3131.02	-3143	-3091.6	-3107.08	-3128
G	-1111	**-1123.1**	**-1138**	-3122	-3140.22	-3160.9	-3118.2	-3134.22	-3174.8

It is clear from these results that the ant algorithm hyper-heuristic is capable of performing better than other established hyper-heuristics. In the full paper we intend to explore the ant algorithm hyper-heuristic more fully, investigating other variations of each stage of the algorithm (including different values of the various constants, different pheromone laying algorithms (e.g. those discussed in [9]), different tour-length selection functions, and the proportion of use of the greedy ants). Our investigation will consider the performance of these hyper-heuristics on the Presentation Scheduling Problem, and will also explore the behaviour of the ant hyper-heuristics in some depth.

Acknowledgements

This work was supported under EPSRC grant reference number GR/N36837/01.
Thank you to Steven Bagley and Matthew Hardy for their advice and help, and Dr. Helen Ashman for the data.

References

[1] Bonabeau, Dorigo & Theralaz, "Swarm Intelligence: From Natural to Artificial Systems", Oxford University Press 1999.

[2] Burke, Kendall, Newall, Hart, Ross & Schulenberg, "Hyper-heuristics: an Emerging Direction in Modern Search Technology", chapter 16 of Handbook of Meta-heuristics (ed. by Glover and Kochenberger), Kluwer 2003, pages 457-474.

[3] Burke, Dror, Kendall, O'Brien, Redrup & Soubeiga, "An Ant Colony Hyper-heuristic for Timetabling Problems", submitted to the Multidisciplinary International Conference on Scheduling: Theory and Applications 2003.

[4] Cowling, Kendall & Soubeiga. "Hyper-heuristics: A Tool for Rapid Prototyping in Scheduling and Optimisation", Proceedings of the 2nd European Conference on EVOlutionary computation for Combinatorial OPtimisation, EvoCop 2002, Springer Lecture Notes in Computer Science, pages 1-10, 2002.

[5] Dorigo, Maniezzo & Colorni. "The Ant System: Optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man and Cybernetic-Part B, Vol. 26, No.1, pages 1-13, 1996.

[6] Gutjahr, "A Graph-based Ant System and its convergence," Future Generation Computer Systems, ISSN 0167-739X, Volume 16 (2000), Number 8, June 2000, pages 873-888.

[7] Kendall, Soubeiga & Cowling. "Choice Function and Random Hyper-heuristics", 4th Asia-Pacific Conference on Simulated Evolution and Learning, SEAL 2002, Vol. 2 (ed. by Wang, Tan, Furuhashi, Kim & Yao), pages 667-671, 2002.

[8] Kennedy & Eberhart with Shi, "Swarm Intelligence", Morgan Kaufmann Publishers, 2001.

[9] Merkle & Middendorf, "Ant Colony Optimization with the Relative Pheromone Evaluation Method", Proceedings of the 2nd European Conference on EVOlutionary computation for Combinatorial OPtimisation, EvoCop 2002, Springer Lecture Notes in Computer Science, pages 325-333, 2002.