Anti-pheromone as a tool for better exploration of search space

Montgomery, James; Randall, Marcus

Published in:
Ant Algorithms - 3rd International Workshop, ANTS 2002, Proceedings

DOI:
10.1007/3-540-45724-0_9

Licence:
Unspecified

Anti-pheromone as a Tool for Better Exploration of Search Space

James Montgomery* and Marcus Randall

School of Information Technology
Bond University, QLD 4229
{jmontgom, mrandall}@bond.edu.au

Abstract. Many animals use chemical substances known as pheromones to induce behavioural changes in other members of the same species. The use of pheromones by ants in particular has lead to the development of a number of computational analogues of ant colony behaviour including Ant Colony Optimisation. Although many animals use a range of pheromones in their communication, ant algorithms have typically focused on the use of just one, a substance that encourages succeeding generations of (artificial) ants to follow the same path as previous generations. Ant algorithms for multi-objective optimisation and those employing multiple colonies have made use of more than one pheromone, but the interactions between these different pheromones are largely simple extensions of single criterion, single colony ant algorithms. This paper investigates an alternative form of interaction between normal pheromone and anti-pheromone. Three variations of Ant Colony System that apply the anti-pheromone concept in different ways are described and tested against benchmark travelling salesman problems. The results indicate that the use of anti-pheromone can lead to improved performance. However, if anti-pheromone is allowed too great an influence on ants' decisions, poorer performance may result.

Keywords: Ant colony optimisation, travelling salesman problem.

1 Introduction

Many animal species, and insects in particular, use chemical substances called pheromones to influence the behaviour of other animals of the same type. Pheromones can carry many different types of information and influence behaviour in varied ways [1]. Many species of ants are known to use pheromones to communicate to coordinate activities like the location and collection of food [2]. The success of this kind of indirect communication has lead researchers to develop a number of simulations of ant behaviour, including optimisation heuristics such as Ant Colony Optimisation (ACO). Based on the foraging behaviour of ant colonies, ACO has generally used a single kind of pheromone to communicate

* This author is a PhD scholar supported by an Australian Postgraduate Award.

between its (artificial) ants, as this is what biological ants do when foraging. However, natural pheromonal communication often consists of a more complex interaction of a number of different pheromones [1]. Furthermore, ACO's reliance on positive feedback alone may make it difficult for it to successfully escape local optima [3,4]. Schoonderwoerd et al. [5] were some of the first to suggest that the use of an "anti-pheromone", the effect of which would be opposite to that of normal pheromone, could be a useful technique in ant algorithms for optimisation. This paper investigates ways in which the concept of an anti-pheromone can be applied to the Travelling Salesman Problem (TSP). Three variations of an Ant Colony System (ACS) that use anti-pheromone in some form are described and compared with a typical implementation of ACS.

An anti-pheromone, or any other variant of the pheromone typically used in ant algorithms, is simply a substance with a different effect to that of "normal" pheromone. Hence, a brief summary of those ant algorithms that have used more than one kind of pheromone is presented here, contrasting these ant algorithms with an approach that uses anti-pheromone. Much of the work in ant algorithms that has used more than one kind of pheromone relates to multiple colony ant systems (e.g. [3,4,6,7,8]). In most of these applications, the interaction between colonies has been relatively simple, the transfer of the best solution from one colony to update the pheromone of another colony [6,7,8].

More complex interaction has been investigated by Kawamura, Yamamoto and Ohuchi, and Kawamura et al. [3,4]. Their Multiple Ant Colony System (MACS) is highly flexible and enables pheromone from one colony to have both positive and negative effects on the behaviour of other colonies. A "negative pheromone effect" is where higher amounts of pheromone on an element actually discourage ants from choosing that element. While the MACS approach is highly flexible, it requires considerable memory and computing resources to maintain the multiple colonies, each with its own pheromone matrix, as well as to calculate the influences between colonies. The anti-pheromone ant algorithms described in this paper are simpler and require less memory and computational resources than MACS, yet can still utilise negative pheromone effects to diversify the search.

The other area in which multiple types of pheromone have been used is in multiple objective optimisation. Mariano and Morales [6] propose an Ant-Q algorithm for solving a multi-objective irrigation problem. Their algorithm, MOAQ, maintains a number of "families," one for each optimisation criterion, which communicate with each other by exchanging the best solution found by each. This is the same kind of information exchange used in multiple colony ant algorithms for single objective optimisation.

Iredi, Merkle and Middendorf [9] propose a different multi colony approach for solving bi-criterion optimisation problems. Every colony maintains two pheromone matrices, tailored to one of the optimisation criteria. Ants within a colony differ in their preference for each pheromone, so that they search different regions of the Pareto-optimal front. The idea of using different pheromones to direct the search in different areas has some merit and is a strong influence on the second anti-pheromone application we describe (see Section 3.2).

This paper is organised as follows. Section 2 has a brief overview of ACS and its governing equations. Section 3 further explains the rationale for using two kinds of pheromone and describes how we adapt ACS to make use of anti-pheromone. Section 4 shows the results of using anti-pheromone on some benchmark TSPs while Section 5 gives the conclusions of this work.

2 ACS

ACO is an umbrella term for a number of similar metaheuristics [10] including the Ant Colony System (ACS) metaheuristic [11]. A brief summary of the equations governing ACS when applied to the Travelling Salesman Problem (TSP) is provided here as it forms the basis for our anti-pheromone modifications described in the next section. The reader is referred to Dorigo and Gambardella [12] and Dorigo, Di Caro and Gambardella [10] for a more in-depth treatment of ACO.

The aim of the TSP is to find the shortest path that traverses all cities in the problem exactly once, returning to the starting city. In a TSP with N cities, the distance between each pair of cities i and j is represented by $d(i, j)$. In ACS, m ants are scattered randomly on these cities ($m \leq N$). In discrete time steps, all ants select their next city then simultaneously move to their next city. Ants deposit pheromone on each edge they visit to indicate the utility (goodness) of these edges. The accumulated strength of pheromone on edge (i, j) is denoted by $\tau(i, j)$.

Ant k located at city r chooses its next city s by applying Equations 1 and 2. Equation 1 is a greedy selection technique favouring links with the best combination of short distance and large pheromone levels. Equation 2 balances this by allowing a probabilistic selection of the next city.

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ \tau(r,u)[d(r,u)]^{\beta} \right\} & \text{if } q \leq q_0 \\ \text{Equation 2} & \text{otherwise} \end{cases} \tag{1}$$

$$p_k(r,s) = \begin{cases} \frac{\tau(r,s)[d(r,s)]^{\beta}}{\sum_{u \in J_k(r)} \tau(r,u)[d(r,u)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Where:

$q \in [0, 1]$ is a uniform random number.
q_0 is the proportion of occasions when the greedy selection technique is used.
$J_k(r)$ is the set of cities yet to be visited by ant k.

The pheromone level on the selected edge (r, s) is updated according to the local updating rule in Equation 3.

$$\tau(r,s) \leftarrow (1-\rho) \cdot \tau(r,s) + \rho \cdot \tau_0 \tag{3}$$

Where:

ρ is the local pheromone decay parameter, $0 < \rho < 1$.

τ_0 is the initial amount of pheromone deposited on each of the edges.

Upon conclusion of an iteration (i.e. once all ants have constructed a tour), global updating of the pheromone takes place. Edges that compose the best solution (over *all* iterations) are rewarded with a relatively large increase in their pheromone level. This is expressed in Equation 4.

$$\tau(r,s) \leftarrow (1-\gamma) \cdot \tau(r,s) + \gamma \cdot \Delta\tau(r,s) \tag{4}$$

$$\Delta\tau(r,s) = \begin{cases} \frac{Q}{L} \text{ if } (r,s) \in \text{ globally best tour} \\ 0 \text{ otherwise.} \end{cases} \tag{5}$$

Where:

$\Delta\tau(r,s)$ is used to reinforce the pheromone on the edges of the global best solution (see Equation 5).
L is the length of the best (shortest) tour to date while Q is a problem dependent parameter [11].
γ is the global pheromone decay parameter, $0 < \gamma < 1$.

3 Anti-Pheromone Applications

As ants construct solutions they gain knowledge of which elements have high utility and which elements may, although desirable in the short-term, lead to poorer solutions in the long term. Randall and Montgomery [13] make use of this in their Accumulated Experience Ant Colony (AEAC) by weighting elements based on their long term effects on solution quality. Anti-pheromone, a substance generally laid down on the elements of poorer solutions, can have a similar effect, making known the accumulated bad experiences of ants that are otherwise lost. This is the approach taken by the first two anti-pheromone algorithms. The third algorithm takes a different approach by making normal pheromone repellent to a small number of ants, rather than depositing anti-pheromone on poorer solutions. It is included here as for those ants that see pheromone as a repellent substance, it represents an anti-pheromone.

3.1 Subtractive Anti-Pheromone (SAP)

As ants construct solutions, they often identify relatively poor solutions as well as good solutions. In this version of the ACS algorithm, particular attention is paid to those poorer solutions, with pheromone being removed from those elements that make up the worst solution in each iteration. Thus, subsequent generations of ants are discouraged from using elements that have formed part of poorer solutions in the past. This constitutes the simplest way to implement anti-pheromone where the deposition of a repellent pheromone is simulated by a reduction in existing pheromone levels, as suggested by Schoonderwoerd et

al. [5] in their conclusions. We refer to this application of anti-pheromone as *Subtractive Anti-Pheromone*.

The SAP algorithm is identical to ACS except for the addition of a second part to the global pheromone update, in which pheromone is removed from links that compose the worst solution in that iteration. This is described in Equation 6.

$$\tau(r,s) \leftarrow \tau(r,s) \cdot \gamma' \qquad \forall (r,s) \in v_w \tag{6}$$

Where:

γ' is the pheromone removal rate due to anti-pheromone.
v_w is the iteration worst solution.

The rate at which pheromone is removed from the elements of the iteration worst solution is controlled by the parameter γ'. The value of γ' used in the experiments is 0.5, which was found to yield the best results.

3.2 Preferential Anti-Pheromone (PAP)

Iredi et al. [9] propose an ant system for solving bi-criterion optimisation problems that uses two types of pheromone, one for each criterion. Their use of a different pheromone for each optimisation criterion allows for knowledge concerning both criteria to be improved as the algorithm progresses. The second anti-pheromone application we propose, called *Preferential Anti-Pheromone*, takes a similar approach by explicitly using two types of pheromone, one for good solutions and one for poorer solutions. Ants in this version of the algorithm differ in their preference for normal pheromone versus anti-pheromone (denoted by τ') with respect to a parameter λ. The value of λ for ant k, $k = [1, m]$, is given by $\frac{k-1}{m-1}$, as in Iredi et al. [9]. Hence, instead of optimising across two objective functions, PAP allows some ants to explore apparently poorer areas of the search space while other ants focus on the solution space near the current global best solution.

Equations 1 and 2 are modified for this variant to incorporate anti-pheromone information, yielding Equations 7 and 8 respectively.

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ [\lambda\tau(r,u) + (1-\lambda)\tau'(r,u)] \cdot [d(r,u)]^\beta \right\} & \text{if } q \leq q_0 \\ \text{Equation 8} & \text{otherwise} \end{cases} \tag{7}$$

$$p_k(r,s) = \begin{cases} \frac{[\lambda\tau(r,s)+(1-\lambda)\tau'(r,s)] \cdot [d(r,s)]^\beta}{\sum_{u \in J_k(r)} [\lambda\tau(r,u)+(1-\lambda)\tau'(r,u)] \cdot [d(r,u)]^\beta} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Pheromone and anti-pheromone are updated equally by ants traversing links during an iteration as local updating ignores the cost of solutions produced. Thus, in addition to Equation 3 being applied, the anti-pheromone on a selected edge is updated by Equation 9.

$$\tau'(r,s) \leftarrow (1-\rho) \cdot \tau'(r,s) + \rho \cdot \tau_0 \tag{9}$$

Upon conclusion of an iteration the global update rule in (4) is applied without modification. In addition, the global update rule for anti-pheromone given in Equation 10 is applied.

$$\tau'(r,s) \leftarrow (1-\gamma)\cdot\tau'(r,s) + \gamma\cdot\Delta\tau'(r,s) \tag{10}$$

$$\Delta\tau'(r,s) = \begin{cases} \frac{Q}{L_w} & \text{if } (r,s) \in \text{ iteration worst tour} \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Where:

$\Delta\tau'(r,s)$ is used to reinforce the pheromone on the edges of the iteration worst solution (see Equation 11).
L_w is the length of the worst tour from the iteration just ended.

3.3 Explorer Ants

The third anti-pheromone variant takes a different approach to the first two in that it does not associate anti-pheromone with poorer solutions. Instead, a small number of ants are chosen to behave differently from other ants by being attracted to areas with little pheromone. These *explorer ants* influence their environment by depositing pheromone in the same way as normal ants, only their preference for existing pheromone is reversed. Hence, an explorer ant finds its own pheromone undesirable. Equations 12 and 13 express how an explorer ant located at city r chooses the next city to go to s.

$$s = \begin{cases} \arg\max_{u\in J_k(r)} \left\{[\tau_{max} - \tau(r,u)]\cdot[d(r,u)]^\beta\right\} & \text{if } q \le q_0 \\ \text{Equation 13} & \text{otherwise} \end{cases} \tag{12}$$

$$p_k(r,s) = \begin{cases} \frac{[\tau_{max}-\tau(r,s)]\cdot[d(r,s)]^\beta}{\sum_{u\in J_k(r)}[\tau_{max}-\tau(r,u)]\cdot[d(r,u)]^\beta} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Where:

τ_{max} is the highest current level of pheromone in the system.

The *explorer ants* algorithm divides the population of ants into two groups, with a higher proportion of normal ants than explorers. We found that two explorer ants produced good results when $m = 10$. While this approach appears similar to the MACS of Kawamura et al. [4], there are some important differences. Although ants are divided into two groups they do not represent separate colonies as they share the same pheromone. Furthermore, this algorithm allows for a small number of explorer ants to be used and saves on memory requirements by using only a single pheromone matrix.

Table 1. TSP instances used in this study

Instance	Description	Optimal Cost
gr24	24 cities	1272
eil51	51 cities	426
eil76	76 cities	538
kroA100	100 cities	21282
d198	198 cities	15780
lin318	318 cities	42029
pcb442	442 cities	50778

4 Computational Experience

A control strategy (normal ACS) and the three alternative implementations were run in order to evaluate their relative performance. Table 1 describes the TSP instances [14] with which the alternative pheromone representations are tested.

The computing platform used to perform the experiments is a 550 MHz Linux machine. The computer programs are written in the C language. Each problem instance is run across 10 random seeds consisting of 3000 iterations. The ACS parameter settings used are: $\beta = -2$, $\gamma = 0.1$, $\rho = 0.1$, $m = 10$, $q_0 = 0.9$.

4.1 Results

The results are given in Table 2. The minimum ("Min"), median ("Med"), maximum ("Max") and inter-quartile range ("IQR") are used to summarise the results as they are non-normally distributed. As the results for CPU time are highly consistent for each combination of algorithm and problem, only the median CPU time (in seconds) is presented in the table.

The results for CPU time are highly consistent with the three anti-pheromone algorithms' times within 3% of those for the control. *Explorer ants* runs slightly slower than the others due to the increased computational overhead associated with evaluating the combined value of two pheromones.

To allow for statistical analysis of cost results across problems, the costs of solutions were normalised according to $\frac{c-c_{opt}}{c_{opt}}$, where c is the cost of the solution and c_{opt} is the optimal cost for its corresponding problem. As the data are non-normally distributed, the Mann-Whitney test was used to compare results.

SAP performs well on problems with less than 100 cities, producing results that are better than the control. This result is statistically significant, $p < 0.05$. On problem `gr24`, it found the optimal solution on every run. SAP also produces better results on `eil51`, and equivalent results on `eil76`, `kroA100` and `d198`. However, the control performs better on problems with more than 200 cities. Across all problems there is no statistically significant difference between the two algorithms. Although in general $\gamma' = 0.5$ yields the best results for SAP on the problem instances tested, other values were investigated. It was found that

Table 2. Results for ACS control and anti-pheromone variants

Problem Instance	Algorithm	Cost				CPU Time
		Min	Med	Max	IQR	
gr24	Control	1272	1278	1278	6	18
	SAP	1272	1272	1272	0	18
	PAP	1272	1272	1278	0	20
	Explorer	1272	1272	1278	5	17
eil51	Control	426	430	441	7	80
	SAP	426	428	430	1	79
	PAP	426	430	436	3	83
	Explorer	426	430	439	4	78
eil76	Control	540	545	554	8	177
	SAP	539	550	558	9	176
	PAP	539	552	562	7	182
	Explorer	539	552	561	11	172
kroA100	Control	21296	21479	22178	371	307
	SAP	21319	21552	22060	382	304
	PAP	21292	21753	22754	411	315
	Explorer	21305	21515	22318	426	298
d198	Control	15948	16116	16451	151	1181
	SAP	15988	16156	16454	337	1183
	PAP	16449	16769	17182	311	1216
	Explorer	16058	16205	16425	169	1161
lin318	Control	45514	46793	47422	1031	3018
	SAP	48375	49099	50608	1026	3050
	PAP	46793	49434	50223	954	3136
	Explorer	45031	46314	48114	908	3027
pcb442	Control	60525	62420	65014	1610	5988
	SAP	62753	64596	66332	1941	5932
	PAP	61623	64156	64753	1133	6097
	Explorer	61709	63657	66663	2219	5883

increasing γ' to 0.75 improves SAP's performance on problems with more than 200 cities, bringing them closer to those achieved by the control. However, there is still a statistically significant difference between SAP with $\gamma = 0.75$ and the control on problems with more than 200 cities, $p < 0.05$.

On a per problem basis, PAP produces worse results than the control on all problems except `gr24` and `eil51`. Analysis across all problems combined reveals a statistically significant difference between the results of the control and those of PAP, $p < 0.10$. It is possible that this poor performance is due to the local update rule in which all ants, regardless of their preference for normal pheromone versus anti-pheromone, update both pheromones by the same amount. Hence, ants with a strong preference for anti-pheromone may distract the search too much and prevent ants with a stronger preference for normal pheromone from searching near the current global best solution. A modified version of PAP was implemented in which ants locally update pheromone in proportion to their value of λ (and anti-pheromone in proportion to $(1 - \lambda)$) producing equivocal results.

Compared across all problems, no statistically significant difference exists between *explorer ants* and the control. Only on `d198` is there sufficient evidence that the control performs better. *Explorer ants* performed slightly better than the control `gr24` and `eil51`.

In general, SAP produces better solutions than PAP. Although PAP performs better than SAP on problems `lin318` and `pcb442`, there is no statistically significant difference. On problems with less than 100 cities, SAP produces better results than *explorer ants*. However, on problems with more than 200 cities, *explorer ants* performs better. *Explorer ants* also performs better than PAP across all problems.

5 Conclusions

Although animals in nature generally use a number of pheromones in their communication, typical ant algorithms have used only one. The majority of ant algorithms that have used more than one pheromone are simple extensions of single-pheromone ant algorithms, maintaining multiple colonies and using the best solution from one colony to update the pheromone of others. More complex pheromone interactions have been used by Kawamura et al. [3,4], but these require fairly considerable computational resources to store and process multiple types of pheromone.

We have proposed three variations of the Ant Colony System that employ anti-pheromone in some form. The first, *subtractive anti-pheromone*, simulates anti-pheromone by reducing the amount of pheromone on elements of the iteration worst solution. It works well on problems with less than 200 cities. It also has a distinct advantage over multiple colony ant systems in that it stores only one kind of pheromone and is no slower than normal ACS. The second algorithm, *preferential anti-pheromone*, is less successful, producing better results than the control on only the two smallest problems. It is possible this is because ants with a strong preference for anti-pheromone distract ants with a preference

for normal pheromone, or that the linear equation for deciding ants' preferences results in too few ants with a strong preference for normal pheromone. *Explorer ants*, the third anti-pheromone algorithm, changes the response to pheromone of a small number of ants, making these ants seek elements with lower pheromone levels. It can produce better solutions than the control on small problems, but produces largely equivalent results on all other problems.

Middendorf, Reischle and Schmeck [8] suggest that in multiple colony ant algorithms the amount and frequency of pheromone information exchange should be kept small. Hence, we plan to extend these algorithms so that the effects of anti-pheromone on normal pheromone are kept to a minimum. For instance, PAP could be improved by having a larger proportion of ants with a strong preference for normal pheromone as well as changing the local update rule so that ants update each kind of pheromone in proportion to their preference for that pheromone. SAP may also yield improved results if pheromone is removed from elements of poor solutions less frequently.

We have given plausible explanations for how each of the three algorithms helps to explore different areas of the search space and why this may prove beneficial. An important extension to this work is to analyse this exploration behaviour more objectively. This will involve measuring the differences in exploration between algorithms as well as the utility of this exploration.

This work is part of a wider strategy that is looking at ways of producing generic strategies to enhance ant based metaheuristics [13,15,16]. In addition to the improvements suggested above, future work will involve the extension of these anti-pheromone applications to other problems.

References

1. Vander Meer, R.K., Breed, M.D., Winston, M.L., Espelie, K.E. (eds.): Pheromone Communication in Social Insects. Ants, Wasps, Bees, and Termites. Westview Press, Boulder, Colorado (1997)
2. Heck, P.S., Ghosh, S.: A Study of Synthetic Creativity through Behavior Modeling and Simulation of an Ant Colony. IEEE Intelligent Systems **15** (2000) 58–66
3. Kawamura, H., Yamamoto, M., Ohuchi, A.: Improved Multiple Ant Colonies System for Traveling Salesman Problems. In Kozan, E., Ohuchi, A. (eds.): Operations Research/Management Science at Work. Kluwer, Boston (2002) 41–59
4. Kawamura, H., Yamamoto, M., Suzuki, K., Ohuchi, A.: Multiple Ant Colonies Algorithm Based on Colony Level Interactions. IEICE Transactions, Fundamentals **E83-A** (2000) 371–379
5. Schoonderwoerd, R., Holland, O.E., Bruten, J.L., Rothkrantz, L.J.M.: Ant-Based Load Balancing in Telecommunications Networks. Adaptive Behavior **2** (1996) 169–207
6. Mariano, C.E., Morales, E.: MOAQ: An Ant-Q Algorithm for Multiple Objective Optimization Problems. Genetic and Evolutionary Computation Conference (GECCO-99), Orlando, Florida (1999) 894–901
7. Michels, R., Middendorf, M.: An Ant System for the Shortest Common Supersequence Problem. In Corne, D., Dorigo, M., Glover, F. (eds.): New Ideas in Optimization. McGraw-Hill, London (1999) 51–61

8. Middendorf, M., Reischle, F., Schmeck, H.: Multi Colony Ant Algorithms. Parallel and Distributed Computing, Proceedings of the 15 IPDPS 2000 Workshops, Third Workshop on Biologically Inspired Solutions to Parallel Processing Problems (BioSP3), Cancun, Mexico (2000) 645–652
9. Iredi, S., Merkle, D., Middendorf, M.: Bi-Criterion Optimization with Multi Colony Ant Algorithms. Evolutionary Multi-Criterion Optimization, First International Conference (EMO'01), Zurich (2001) 359–372
10. Dorigo, M., Caro, G.D.: The Ant Colony Optimization Meta-heuristic. In Corne, D., Dorigo, M., Glover, F. (eds.): New Ideas in Optimization. McGraw-Hill, London (1999) 11–32
11. Dorigo, M., Gambardella, L.M.: Ant Colonies for the Traveling Salesman Problem. BioSystems **43** (1997) 73–81
12. Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant Algorithms for Distributed Discrete Optimization. Artificial Life **5** (1999) 137-172
13. Randall, M., Montgomery, J.: The Accumulated Experience Ant Colony for the Travelling Salesman Problem. Proceedings of Inaugural Workshop on Artificial Life, Adelaide, Australia (2001) 79–87
14. Reinelt, G.: TSPLIB - A Traveling Salesman Problem Library. ORSA Journal of Computing **3** (1991) 376–384
15. Montgomery, J., Randall, M.: Alternative Pheromone Applications for Ant Colony Optimisation. Technical Report TR02-07, School of Information Technology, Bond University, Qld, Australia. Submitted to AI2002, 15th Australian Joint Conference on Artificial Intelligence, Canberra, Australia (2002)
16. Randall, M.: A General Framework for Constructive Meta-heuristics. In Kozan, E., Ohuchi, A. (eds.): Operations Research/Management Science at Work. Kluwer, Boston, MA (2002) 111–128