Intensification and diversification strategies in ant colony system

Randall, Marcus; Tonkcs, Elliot

Published in:
Complexity International

Published: 01/12/2002

Document Version:
Publisher's PDF, also known as Version of record

COMPLEXITY INTERNATIONAL

Volume 09
http://www.complexity.org.au/vol09/

Intensification and Diversification Strategies in Ant Colony System

Marcus Randall and Elliot Tonkes
School of Information Technology
Bond University, QLD 4229

August 23, 2002

Abstract

The ant colony optimisation class of meta-heuristics simulate the ability of ant colonies to determine shortest paths to food. In this paper, we investigate strategies for achieving a balance between intensification and diversification search phases apart from that implicit in the algorithm itself. This is achieved by continually varying the sensitivity with which ants(agents) perceive pheromone trails. This has the effect of allowing the ants to both refine the search in promising regions and to escape explored areas of the search space. We examine four strategies using the well-known travelling salesman problem. The results indicate that while the explicit intensification/diversification schemes occasionally outperform the standard ant colony optimisation, the algorithm's inherent intensification/diversification characteristics are often sufficient to find good quality solutions.

Keywords: Ant colony optimisation, Ant colony system, combinatorial optimisation, meta-heuristic, travelling salesman problem.

1 Introduction

Ant Colony Optimisation (ACO) [7] is a relatively new class of meta-heuristic search techniques. The meta-heuristics simulate the ability of ant colonies to determine shortest paths to food and is based on research into natural ants [13]. Although individual ants (also known as *agents*) possess few capabilities, their operation as a colony is capable of complex behaviour. ACO techniques expand upon existing paradigms which simulate natural behaviour such as neural networks, simulated annealing (SA) and genetic algorithms (GAs). In this paper we concentrate on one of the ACO methods, namely Ant Colony System (ACS) [9].

Various ACO techniques have been applied to a wide variety of optimisation problems including vehicle routing, graph colouring and network routing [8]. Extensive benchmarking of ant systems has utilised the travelling salesman problem (TSP). In this paper, we augment the ACS meta-heuristic with various intensification and diversification strategies to complement its own search techniques. Intensification and diversification strategies have been successfully implemented in the Tabu Search (TS) meta-heuristic. According to Glover and Laguna [11], TS becomes a much stronger and more robust technique when strategies such as intensification and diversification are applied.

The remainder of this paper is organised as follows. Section 2 describes the workings of ACS while Section 3 explains the intensification and diversification strategies that we use. Section 4 describes our computational experiments on a range of standard TSP instances. Finally, Section 5 summarises the results and outlines our future work.

2 ACS

ACS can best be described with the TSP metaphor. Consider a set of cities, with known distances between each pair of cities. The aim of the TSP is to find the shortest path to traverse all cities exactly once and return to the starting city. The ACS paradigm is applied to this problem in the following way. Consider a TSP with N cities. Cities i and j are separated by distance $d(i,j)$. Scatter m virtual ants randomly on these cities ($m \le N$). In discrete time steps, all ants select their next city then simultaneously move to their next city. Ants deposit a substance known as 'pheromone' to communicate with the colony about the utility (goodness) of the edges. Denote the accumulated strength of pheromone on edge (i,j) by $\tau(i,j)$.

At the commencement of each time step, Equations 1 and 2 are used to select the next city s for ant k currently at city r. Note that $q \in [0,1]$ is a uniform random number and q_0 is a parameter. To maintain the restriction of unique visitation, ant k is prohibited from selecting a city which it has already visited. The cities which have not yet been visited by ant k are indexed by $J_k(r)$.

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

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

It is typical that the parameter β is negative so that shorter edges are favoured. Linear dependence on $\tau(r,s)$ ensures preference is given to links that are well traversed (i.e. have a high pheromone level). Equation 1 is a highly greedy selection technique favouring cities which possess the best combination of short distance and large pheromone levels. Equation 2 balances this by allowing a probabilistic selection of the next city.

The pheromone level on the selected edge 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 small amount of pheromone deposited on each of the edges. According to [9], a good initial pheromone is $\tau_0 = (NL_{nn})^{-1}$ where L_{nn} is the cost produced by the nearest neighbour heuristic.

Upon conclusion of an iteration (i.e. all ants have constructed a tour), global updating of the pheromone takes place. The edges that compose the best solution to date[1] are rewarded with an increase in their pheromone level. This is expressed in Equation 4.

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

Where:

$\Delta\tau(r,s)$ is used to enforce the pheromone on the edges of the solution (see Equation 5). L is the length of the best (shortest) tour to date while Q is a constant that is set to 100 [10].

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

γ is the global pheromone decay parameter, $0 < \gamma < 1$.

When the problem size becomes large, ACS can require significant computational time to process each iteration. At each step, the list of feasible cities is large. In order to reduce this time, Dorigo and Gambardella [9] suggest the use of *candidate sets*. A candidate set stores a list of the *cl* closest cities to each city in the problem (typically $cl \ll N$). The ants use this set to select the next city to visit, significantly reducing the overall computation time. Only if the elements of this set lead to an infeasible solution will the ant consider the remainder of the cities.

The basic operation of ACS is described by the pseudocode in Figure 1.

```
Initialise pheromone on all edges;
While (stopping criterion is not met)
  Deposit each ant on a random city such that no two
  ants are placed on the same city;
  For(the number of cities)
    For(each ant)
      Choose the next city to visit according to
      Equation 1;
    End For;
    For(each ant)
      Update the pheromone on each edge according
      to Equation 3;
    End For;
  End For;
  If the best tour from this iteration is better than the
  globally best tour Then set this as the globally best
  tour;
  Reinforce the pheromone of the edges belonging to the
  globally best tour according to Equation 4;
End While;
Output the globally best tour and cost;
```

Figure 1: Pseudocode of ACS applied to the TSP.

[1]This is known as the *global-best* [9] scheme. An *iteration-best* scheme, where the edges of the best solution in the current colony of ants is used, is also possible.

3 Achieving Intensification and Diversification

Many advanced search processes such as TS and GRASP [11] implement different search phases in order to achieve good quality solutions. One such approach involves the use of *intensification* and *diversification* strategies. Intensification aims to identify solution attributes that are common to good solutions and to encourage the search process to seek solutions with those attributes. Diversification is complementary to this, as it allows the search process to enter unexplored regions of the state space.

Intensification and diversification techniques are present to an extent in the existing ACS algorithm. Diversification is achieved by the application of the local pheromone update rule (Equation 3) as pheromone evaporates from the visited edges, thus encouraging the system to incorporate less-used edges in subsequent iterations. Simultaneously, the pheromone on the edges of the globally best solution is reinforced by Equation 4. However, if many ants choose to incorporate an edge having a high pheromone level, once again the local pheromone decay effect will ensure that its future use is discouraged.

Our aim in this paper is to provide ACS with explicit intensification and diversification strategies that are beyond those already present in the algorithm. To do this we reintroduce the α parameter from the original ant system [10] into Equations 1 and 2. This parameter governs the relative importance of the collective pheromone trails in the selection of the next city. Equations 6 and 7 are the modified forms of Equations 1 and 2 respectively.

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{[\tau(r,s)]^{\alpha} \cdot [d(r,s)]^{\beta}\right\} & \text{if } q \leq q_0 \\ \text{Equation 7} & \text{otherwise} \end{cases} \tag{6}$$

$$p_k(r,s) = \begin{cases} \frac{[\tau(r,s)]^{\alpha}[d(r,s)]^{\beta}}{\sum_{u \in J_k(r)} [\tau(r,u)]^{\alpha}[d(r,u)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In Equations 6 and 7, small values of α imply that the influence of the deposited pheromone is small. For $\alpha < 0$, the system will actively discourage the use of edges with large amounts of deposited pheromone. This corresponds to diversification, as edges that have been used heavily in the past will have a relatively small chance of being chosen, thus encouraging the use of other edges. Conversely for $\alpha \geq 1$, the level of pheromone is considerably more important and as such, edges that have been frequently incorporated into ant tours will be favoured. This corresponds to an intensification strategy.

In previous works [3, 9, 10], α is held constant. However, in this study we will vary α throughout the search process in order to achieve intensification and diversification phases. Some traditional ant system techniques [3, 4, 6] have been documented to suffer failure due to *stagnation behaviour* (especially if certain parameter settings are used) [10]. When a balance is not achieved between exploring new sections of the search space and reinforcing existing solutions, high concentrations of pheromone appear on a single path. Further iterations simply force ants to follow the same path, produce identical tours and thus the method stagnates. This is similar to *premature convergence* in GAs [12]. Varying α provides a systematic approach to deal with possible stagnation of solutions, and to effectively sample a broader region of solution space. Colorni, Dorigo, Maffioli, Maniezzo, Righini and Trubian [4] have attempted to overcome stagnation by trail modification and genetic modification of α and β (on a trial by trial basis) with moderate success.

We consider three approaches for the systematic variation of α and a control strategy. Index the iteration by x and consider α as a function of x, where $0 \leq x \leq 2500$ (the maximum number of iterations) and $-0.5 \leq \alpha \leq 1.5$.

- **S0:** $\alpha = 1$ for the duration of the search. This corresponds to traditional ACS and represents the control strategy.
- **S1:** As the algorithm progresses, the magnitude of α is slowly increased from -0.5 to 1.5. This proposal is motivated by the SA philosophy [15]. In early stages of SA, the temperature is high and solutions experience considerable variation (corresponding to small values of α). As the temperature cools, solutions are less likely to deviate and a process of refinement takes over (corresponding to large values of α). This is expressed by Equation 8.

$$\alpha_{S1}(x) = \frac{x}{1250} - 0.5 \tag{8}$$

- **S2:** A *reheating-like* approach commonly used in SA [1, 5] is applied to α. This method periodically decreases the level of α if no improvement in the overall best cost has been detected within a given period. If this is the case, α is reset to a level half-way between the last reheating point and the current value for α. Between these resets, α follows Equation 8 from method S1. The number of iterations between reheats (denoted rh) is a parameter of the process.
- **S3:** The *strategic oscillation* approach from TS [11] is applied to α. By oscillating between intensification and diversification phases, the method is expected to sample the solution space. This is expressed by Equation 9.

$$\alpha_{S3}(x) = \sin\left(1 + \frac{x}{50}\right) + 0.5 \tag{9}$$

Graphical representations of α for schemes S0, S1 and S3 are shown in Figure 2(a). Figure 2(b) shows an example of S2.

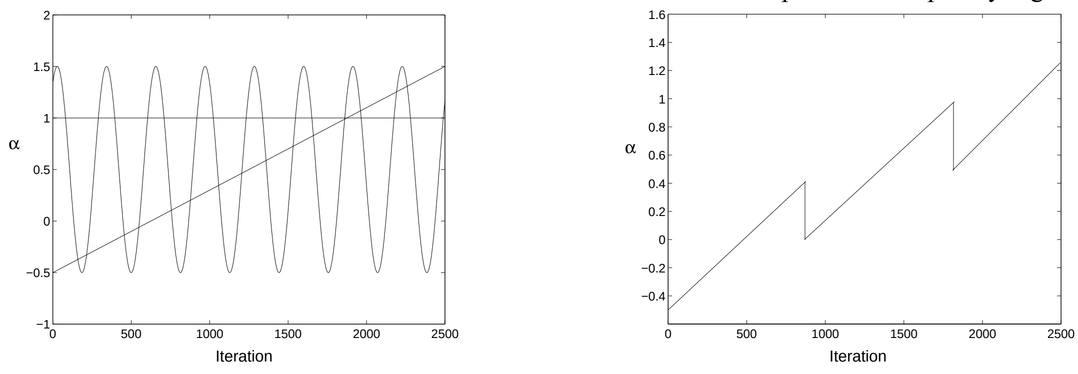

Figure 2: (a) The variation of α for the S0, S1 and S3 schemes over 2500 iterations. (b) Example of a typical run of S2 displaying the variation of α over 2500 iterations.

Table 1: Problem instances used in this study.

Name	Size (cities)	Best-Known Cost
gr24	24	1272
hk48	48	11461
eil51	51	426
st70	70	675
eil76	76	538
kroA100	100	21282
ch130	130	6110
d198	198	15780
kroA200	200	29368
lin318	318	42029
pcb442	442	50778
rat575	575	6773
d657	657	48912

4 Computational Experience

We have selected a number of TSP problem instances with which to test the ACS engine. These problems are from TSPLIB [14] and are given in Table 1. In this paper, we divide these problems into two groups: *the small TSPs* (gr24 - kroA100) and *the large TSPs* (ch130 - d657). For the large problems, we implement the candidate set technique in order to reduce the amount of computational time per run.

These problems are run using the set of parameters given in Table 2 as these have been found to give good performance in [9, 10]. rh was chosen after some initial experimentation. The computer platform used to perform the experiments is an IBM SP2 consisting of 22 RS6000 model 590 processors with a peak performance of 266 MFLOPS per node. Each problem is run using 10 random seeds.

Table 3 shows the results of running the S0, S1, S2 and S3 schemes on the small TSPs. The results table is divided into two sections, **Cost** and **Runtime**. For each section, the minimum (**Min**), average (μ), maximum (**Max**) and standard deviation (σ) are used. The **Runtime** column records the number of CPU seconds to achieve the best cost for a particular run.

It is evident from Table 3 that S0 consistently obtains better solution costs than the other schemes. However, it only marginally outperforms S2. S3 performs poorly against the other schemes on the

Table 2: Parameter settings used in this study.

Parameter	Value
α	1 (for S0)
β	-2
γ	0.1
ρ	0.1
m	10
q_0	0.9
rh	300
cl	15 (for large problems)
iterations	2500

Table 3: Results of ACS for the small TSPs.

Scheme	Problem	Best	Cost				Runtime	(seconds)		
		Known Cost	Min	μ	Max	σ	Min	μ	Max	σ
S0	gr24	1272	1272	1276.2	1278	2.9	0.04	7.38	35.82	11.25
	hk48	11461	11461	11493.6	11578	46.39	54.7	105.74	232.4	60.46
	eil51	426	426	430.5	436	3.69	6.27	134.53	254.91	81.03
	st70	675	677	686.9	701	7.28	70.03	299.72	570.1	185.88
	eil76	538	538	547.5	558	7.74	184.87	384.67	583.24	164.36
	kroA100	21282	21319	21660	22443	353.83	268.54	690.96	1241.91	336.87
S1	gr24	1272	1272	1276.2	1278	2.9	14.56	21.45	45.01	10.2
	hk48	11461	11461	11468.2	11470	3.79	87.8	111.75	152	22.03
	eil51	426	428	430	434	1.89	130.26	198.27	250.32	42.2
	st70	675	678	685	699	6.57	376.08	442.81	468.15	27.14
	eil76	538	542	547.7	556	4.64	453.14	530.29	576.57	37.09
	kroA100	21282	21373	21849.2	22722	494.64	777.2	1030.14	1226.37	137.53
S2	gr24	1272	1272	1275.6	1278	3.1	18	33.32	53.53	11.5
	hk48	11461	11461	11475.6	11553	27.52	105.8	182.64	225.9	40.67
	eil51	426	426	430.3	438	3.74	144.26	213.02	262.75	47.23
	st70	675	677	690.8	702	7.89	360.93	418.7	468.15	39.24
	eil76	538	543	548.9	553	3.45	391.7	576.02	686.1	111.35
	kroA100	21282	21389	21629.78	21915	193.03	867.79	1144.72	1325.53	139.22
S3	gr24	1272	1272	1275	1278	3.16	0.07	15.5	49.06	19.68
	hk48	11461	11470	11487.9	11566	36.01	4.7	52.84	218.2	66.45
	eil51	426	427	432	443	5.56	2.69	93.53	213.7	70.96
	st70	675	691	708.2	725	11.89	76.46	306.17	521.42	160.13
	eil76	538	544	557.4	569	7.68	16.12	311.08	631.89	198.2
	kroA100	21282	21512	21900.1	22265	223.38	65.39	378.77	876.9	285.04

small TSPs.

Table 4 shows the results of running the schemes on the large problem set. The schemes varying α perform better than S0 on the larger problems than on the smaller problems. Each scheme provides better solutions than S0 for pcb442 and d657.

Table 4: Results of ACS for the large TSPs.

Scheme	Problem	Best	Cost				Runtime	(seconds)		
		Known Cost	Min	μ	Max	σ	Min	μ	Max	σ
S0	ch130	6110	6251	6315.6	6418	49.12	172.52	316.72	379.24	66.83
	d198	15780	16040	16418.2	16929	259.83	624.72	785.84	899.46	106.08
	kroA200	29368	29572	29983.9	30648	375.28	422.14	743.68	966.51	179.87
	lin318	42029	43690	44246.9	44770	345.7	1372.42	1798.12	1883.14	155.29
	pcb442	50778	56193	57815.7	60209	1436.85	1161.6	3741.5	4705.92	1178.04
	rat575	6773	7034	7191.2	7308	88.42	4692.7	4874.42	4999.92	117.3
	d657	48912	56276	57548.5	59116	947.4	4737.6	9385.44	11716.8	2508.58
S1	ch130	6110	6191	6286.7	6374	62.62	183.31	293.19	379.7	76.5
	d198	15780	16240	16560.4	16927	243.95	629.66	740.13	849.3	68.42
	kroA200	29368	29612	29991.6	30927	374.68	703.44	765.02	819.07	34.49
	lin318	42029	44051	44964.7	45965	627.35	1357.82	1454.98	1598.21	77.9
	pcb442	50778	54549	56339.1	58174	1053.29	3166.08	3403.78	3569.28	143.17
	rat575	6773	7243	7462.8	7826	182.57	4046.12	4258.97	4473.82	140.87
	d657	48912	53775	56479	58147	1226.75	5961.6	8006.88	8884.8	774.13
S2	ch130	6110	6216	6300.2	6387	49.09	252.32	333.79	374.22	42.96
	d198	15780	16186	16456.3	16716	160.59	838.66	907.82	948.48	39.86
	kroA200	29368	29652	30278.1	31142	545.18	759.32	865.2	948.66	69.79
	lin318	42029	43797	44840.7	45800	705.82	1675.78	1774.16	1909.25	75.72
	pcb442	50778	55529	56644.7	57729	750.37	3273.6	4495.49	4798.08	471.58
	rat575	6773	7260	7441	7818	223.57	4582.26	4861.97	5001.93	134.86
	d657	48912	54357	56536	59384	1356.8	10272	11333.76	11995.2	706.66
S3	ch130	6110	6507	6587.7	6659	57.22	11.86	150.53	373	125
	d198	15780	16229	16484.9	16953	287.09	111.72	378.9	948.1	288.53
	kroA200	29368	29627	29897.8	30907	373.23	97.78	694.44	1266.82	332.23
	lin318	42029	43796	45284.1	46455	925.56	817.92	1306.37	1744.9	341.7
	pcb442	50778	54530	55014.9	55631	358.32	1689.6	3282.43	4723.2	1157.64
	rat575	6773	7156	7515.1	7697	166.82	658.62	2704.78	4923.62	1607.69
	d657	48912	55749	56489	57186	494.54	345.6	7355.04	11865.6	4586.19

Figures 3 and 4 show graphically the best and average performance of the ACS schemes respectively. The deviation from the best known solution cost is expressed as the *Relative Percentage Deviation* (RPD) defined by Equation 10.

$$RPD = \frac{cost - opt}{opt} \tag{10}$$

Where *cost* is the solution cost and *opt* is the optimal cost.

5 Conclusions

This paper has documented the application of sinusoidal, linear and restarting/reheating variations in α as iterations progress. These techniques are applicable across a range of ACO meta-heuristics. However, of all the schemes tested, the one in which α is held constant provides the most consistent performance. This scheme corresponds to the ACS outlined in Dorigo and Gambardella [9]. Only occasionally do the schemes varying α perform better in terms of the best and average costs achieved. As pointed out in Section 3, ACS has some implicit intensification and diversification characteristics within the algorithm. Given the results of this study, these appear sufficient on their own to produce good quality solutions. Clearly though, a much wider range of functions could be investigated and these may improve upon traditional ACS. Another possible way to induce intensification and diversification is to vary the β parameter (by itself or in conjunction with α). The authors are in the process of developing more intelligent reheating schemes to complement this work and that done by Colorni et al. [4]. In particular, we wish to investigate schemes that are able to sense the degree of intensification or diversification required during phases of the search process. It is believed that this will yield improved performance of the algorithm.

As future work, a further possibility is that exploration factors in the algorithm can be enhanced by endowing several of the agents in the system with a more random transition probability function

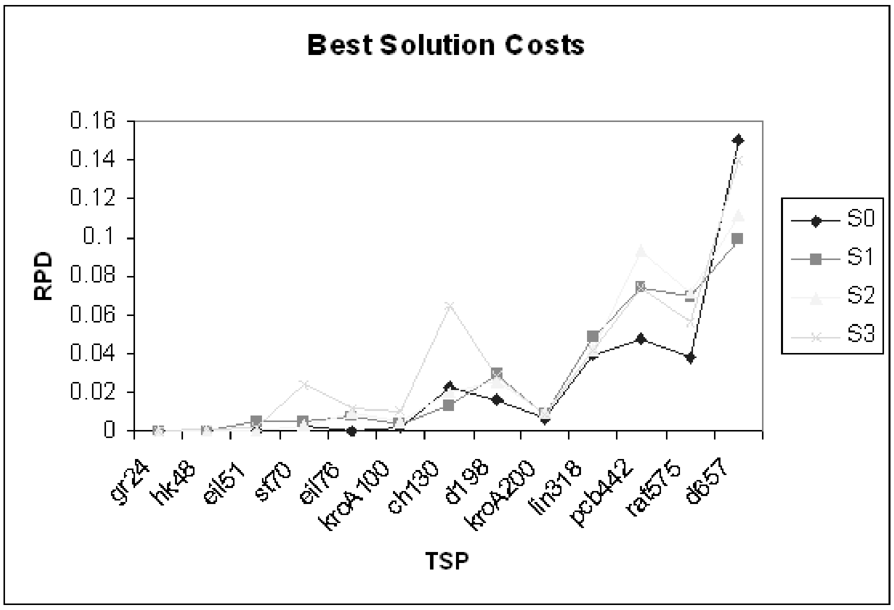

Figure 3: A comparison of the best costs of the schemes on each TSP.

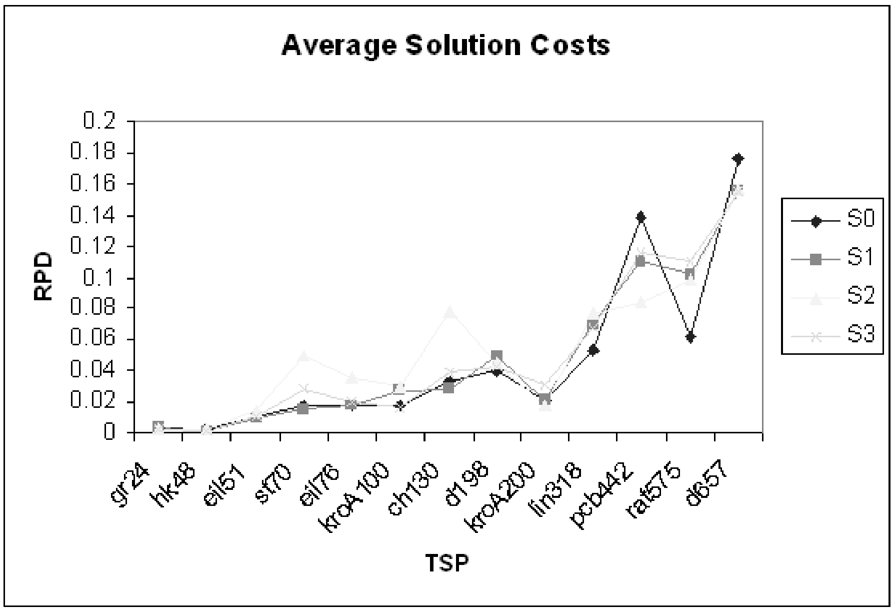

Figure 4: A comparison of the average costs of the schemes on each TSP.

$p_k(r, s)$ and/or changing the local pheromone updating rule (Equation 3). Little attention has been paid to the possibility that agents possess nonuniform properties.

References

[1] Abramson, D. and Randall, M. (1999) "A Simulated Annealing Code for General Integer Linear Programs", Annals of Operations Research, 86, pp. 3-21.

[2] Bullnheimer, B., Hartl, R. and Strauß, C. (1999) "Applying the Ant System to the Vehicle Routing Problem", in Voß, S. et al. (eds.), Meta Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer, Boston.

[3] Colorni, A., Dorigo, M. and Maniezzo, V. (1991) "Distributed Optimization by Ant Colonies", Proceedings of ECAL91, Elsevier, pp. 134-142.

[4] Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G. and Trubian, M. (1992) "Heuristics from Nature for Hard Combinatorial Problems", International Transactions on Operations Research, 3, pp. 1-21.

[5] Connolly, D. (1992) "General Purpose Simulated Annealing", Journal of the Operational Research Society, 43, pp. 495-505.

[6] Dorigo, M. (1992) Optimization, Learning and Natural Algorithms, PhD Thesis, Politecnio di Milano, Italy, 140 pages.

[7] Dorigo, M. and Di Caro, G. (1999) "The Ant Colony Meta-heuristic", in New Ideas in Optimization, Corne, D., Dorigo, M. and Golver, F. (eds), McGraw-Hill, pp. 11-32.

[8] Dorigo, M., Di Caro, G. and Gambardella, L. (1999) "Ant Algorithms for Discrete Optimization", Artificial Life, 5, pp. 137-172.

[9] Dorigo, M. and Gambardella, L. (1997) "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE Transactions on Evolutionary Computation, 1, pp. 53-66.

[10] Dorigo, M., Maniezzo, V. and Colorni, A. (1996) "The Ant System: Optimization by a Colony of Cooperating Agents", IEEE Transactions on Systems, Man and Cybernetics - Part B, 26, pp. 29-41.

[11] Glover, F. and Laguna, M. (1997) Tabu Search, Kluwer Academic Publishers, Boston MA, 402 pages.

[12] Goldberg, D. (1989) Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Reading MA, 412 pages.

[13] Grassé, P. (1959) "La reconstruction du nid et les coordinations inter-individuelles chez *Bellicositermes natalesensis et Cubitermes sp.* La théorie del la stigmergie: Essai d'interprétation des termites constructeurs", Insect Sociaux, 6, pp. 41-83.

[14] Reinelt, G. (1991) "TSPLIB - a Traveling Salesman Problem Library", ORSA Journal on Computing, 3, pp. 376-384. Available on-line at `http://softlib.rice.edu/softlib/tsplib`.

[15] van Laarhoven, L. and Aarts, E. (1987) Simulated Annealing: Theory and Applications, D Reidel Publishing Company: Dordecht, 186 pages.