Particle Swarm for the Traveling Salesman Problem

Elizabeth F. Gouvêa Goldbarg, Givanaldo R. de Souza, and Marco César Goldbarg

Department of Informatics and Applied Mathematics,
Federal University of Rio Grande do Norte,
Campus Universitário Lagoa Nova, Natal , Brazil
{beth, gold}@dimap.ufrn.br, givanaldo@yahoo.com.br

Abstract. This paper presents a competitive Particle Swarm Optimization algorithm for the Traveling Salesman Problem, where the velocity operator is based upon local search and path-relinking procedures. The paper proposes two versions of the algorithm, each of them utilizing a distinct local search method. The proposed heuristics are compared with other Particle Swarm Optimization algorithms presented previously for the same problem. The results are also compared with three effective algorithms for the TSP. A computational experiment with benchmark instances is reported. The results show that the method proposed in this paper finds high quality solutions and is comparable with the effective approaches presented for the TSP.

1 Introduction

The Traveling Salesman is a classical NP-hard combinatorial problem that has been an important test ground for many algorithms. Given a graph G = (N,E), where N = {1,...,*n*} is the set of nodes and E = {1,...,*m*} is the set of edges of G, and costs, c_{ij}, associated with each edge linking vertices *i* and *j*, the problem consists in finding the minimal total length Hamiltonian cycle of G. The length is calculated by the summation of the costs of the edges in a cycle. If for all pairs of nodes {*i*,*j*}, the costs c_{ij} and c_{ji} are equal then the problem is said to be symmetric, otherwise it is said to be asymmetric. A survey of the TSP is presented by Gutin and Punnen [10].

The term A-Life is utilized to describe researches on systems that simulate essential properties of the real life, with two general lines:

- how computational techniques may help the investigation of natural phenomena, and
- how biological techniques may help to solve computational problems.

Several bio-inspired methods were proposed to solve Combinatorial Optimization problems, such as Genetic Algorithms [12], Memetic Algorithms [17], Cultural Algorithms [24] and Ant Systems [5], among others. Particle swarm optimization, PSO, algorithms belong also to the class of bio-inspired methods [14]. This is a population-based technique introduced by a Psychologist, James Kennedy, and an Electrical Engineer, Russell Eberhart, who based their method upon the behavior of bird flocks. PSO algorithms for the TSP were presented previously by Onwubulu and Clerc [18] and Wang et al. [26]. Hybrid PSO approaches for the same problems were also presented by Machado and Lopes [16] and Pang et al. [19].

J. Gottlieb and G.R. Raidl (Eds.): EvoCOP 2006, LNCS 3906, pp. 99 – 110, 2006.

This paper presents a new PSO algorithm for the Traveling Salesman Problem. The main difference between the approach reported in this paper and the previous ones regards the velocity operators. In this paper local search and path-relinking procedures are proposed as velocity operators for a discrete optimization problem.

Local search is a traditional optimization method that starts with an initial solution and proceeds by searching for a better solution in a neighborhood defined for the starting solution. If a better solution is found, then it is assumed as the new current solution and the process of searching in its neighborhood is re-started. This process continues until no improvement of the current solution is found [1].

Path-relinking is an intensification technique which ideas were originally proposed by Glover [8] in the context of scheduling methods to obtain improved local decision rules for job shop scheduling problems [9]. The strategy consists in generating a path between two solutions creating new solutions. Given an origin, x_s, and a target solution, x_t, a path from x_s to x_t leads to a sequence x_s, $x_s(1)$, $x_s(2)$, …, $x_s(r) = x_t$, where $x_s(i+1)$ is obtained from $x_s(i)$ by a move that introduces in $x_s(i+1)$ an attribute that reduces the distance between attributes of the origin and target solutions. The roles of origin and target can be interchangeable. Some strategies for considering such roles are:

- forward: the worst among x_s and x_t is the origin and the other is the target solution;
- backward: the best among x_s and x_t is the origin and the other is the target solution;
- back and forward: two different trajectories are explored, the first using the best among x_s and x_t as the initial solution and the second using the other in this role;
- mixed: two paths are simultaneously explored, the first starting at the best and the second starting at the worst among x_s and x_t, until they meet at an intermediary solution equidistant from x_s and x_t.

The paper is organized as follows. Particle swarm optimization is described in Section 2. The proposed algorithm and its variants are described in Section 3. A computational experiment is reported in Section 4. The experiment compares the results of the proposed approach with previous PSO algorithms. To date PSO algorithms for the TSP have failed to produce results comparable to competitive techniques. A comparison with three effective techniques proposed for the Traveling Salesman Problem shows that the proposed approach produces high quality solutions. Finally, some concluding remarks are presented in Section 5.

2 Particle Swarm Optimization

Particle swarm optimization, PSO, is an evolutionary computation technique inspired in the behavior of bird flocks, fish schools and swarming theory. PSO algorithms were first introduced by Kennedy and Eberhart [14] for optimizing continuous nonlinear functions. The fundamentals of their method lie on researches on computer simulations of the movements of social creatures [11], [21], [23].

Given a population of solutions (the swarm) for a given problem, each solution is seen as a social organism, also called particle. The method attempts to imitate the behavior of the real creatures making the particles “fly” over a solution space, thus balancing the efforts of search intensification and diversification. Each particle has a

value associated with it. In general, particles are evaluated with the objective function being optimized. A velocity is also assigned to each particle in order to direct the "flight" through the problem space. The artificial creatures have a tendency to follow the best ones among them. In the classical PSO algorithm, each particle

- has a position and a velocity
- knows its own position and the value associated with it
- knows the best position it has ever achieved, and the value associated with it
- knows its neighbors, their best positions and their values

As pointed by Pomeroy [20], rather than exploration and exploitation what has to be balanced is individuality and sociality. Initially, individualistic moves are preferable to social ones (moves influenced by other individuals), however it is important for an individual to know the best places visited by its neighbors in order to "learn" good moves.

The neighborhood may be physical or social [18]. Physical neighborhood takes distances into account, thus a distance metric has to be established. This approach tends to be time consuming, since each iteration distances must be computed. Social neighborhoods are based upon "relationships" defined at the very beginning of the algorithm.

The move of a particle is a composite of three possible choices:

- To follow in its own way
- To go back to its best previous position
- To go towards its best neighbor's previous position or towards its best neighbor

Equations (1) and (2) are utilized to update the particle's position and velocity at each time step [6].

$$x_{t+1} = x_t + v_t \tag{1}$$

$$V_{t+1} = w.v_t + c_1.\text{rand}_1.(pbest_t - x_t) + c_2.\text{rand}_2 .(gbest_t - x_t) \tag{2}$$

Where x_t and v_t are the particle's position and velocity at instant t, respectively, $pbest_t$ is the particle's best previous position, $gbest_t$ is the best position that any particle has achieved so far, w is the inertia factor [25], c_1 and c_2 are social/cognitive coefficients that quantify how much the particle trusts its experience and how much it trusts the best neighbor, rand_1 and rand_2 are randomly generated numbers.

To apply PSO to discrete problems, one is required to define a representation for the position of a particle and to define velocity operators regarding the movement options allowed for the particles (and ways for, possibly, combining movements).

A general framework of a PSO algorithm for a minimization problem is listed in the following.

```
procedure PSO
{
  Initialize a population of particles
  do
   for each particle p
```

```
    Evaluate particle p
    If the value of p is better than the value of pbest
       Then, update pbest with p
   end_for
   Define gbest as the particle with the best value
   for each particle p do
    Compute p's velocity
    Update p's position
   end_for
  while (a stop criterion is not satisfied)
}
```

3 PSO for the Traveling Salesman Problem

Usually, evolutionary algorithms represent TSP solutions as a permutation on the n vertices of a given instance. This representation is also adopted in this work. A social neighborhood containing only the global best particle (representing the best current solution) is defined for each member of the population. A particle has three movement options:

- To follow in its own way
- To go back to its best previous position
- To go towards the global best solution (particle)

The main difference between the proposed algorithm and the previous approaches lies on the way the velocity operators are defined. The first option of a particle, that is to follow its own way, is implemented by means of a local search procedure. Two local search procedures were utilized, each of them defining a version of the algorithm. The first local search procedure is based upon an inversion operator to build neighbor solutions. The inversion operator inverts the elements of a particle between two indices, a and b. The difference b-a varies between 1 (simulating a 2-swap move) and n-1. Thus, when v_1 is applied to a particle p, the local search procedure starts inverting sequences of two elements, then inverts sequences of three elements, and so on. Figure 1 illustrates an inversion of a sequence of elements defined by the interval a=2 and b=5 (a sequence with four elements) of particle p resulting on particle p'.

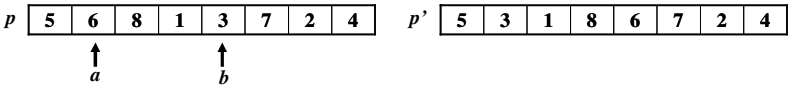

Fig. 1. Inversion of elements among indices a and b

The second local search procedure is the Lin-Kernighan neighborhood, LK [15]. This is a recognized efficient improvement method for the TSP. The basic LK algorithm has a number of decisions to be made and depending on the strategies adopted by programmers distinct implementations of this algorithm may result on different performances. The literature contains reports of many LK implementations with widely varying behavior [13]. In this work the algorithm utilized the LK implementation of the Concorde solver [3].

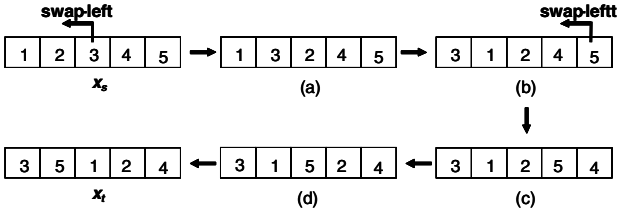

Fig. 2. Path-relinking operator

Another velocity operator is considered when a particle has to move from its current position to another (pbest or gbest). In these cases the authors consider that a natural way to accomplish this task is to perform a path-relinking operation between the two solutions.

The velocity utilized to move a particle from an origin to a target position is defined as a path-relinking operation. The relinking operator utilized in this work is illustrated in Figure 2. The operator swaps one element with its right (left) neighbor. The steps of a path-relinking procedure that explores solutions in the path between the origin solution (1 2 3 4 5) and the target solution (3 5 1 2 4) is shown. First, the element 3 is moved to the first position by swapping it with elements 2 (Figure 2(a)) and 1 (Figure 2(b)). At this point, element 5 has to be moved to the second position. It is swapped with element 4 (Figure 2(c)), element 2 (Figure 2(d)) and, finally, it is swapped with element 1, when the target solution is reached. The swap operators lead to $O(n^2)$ procedures.

The path-relinking is applied simultaneously from the origin to the target solution and vice-versa (back and forward). It is also utilized the swap-left and swap-right operations.

Combinations of movements are possible. For instance, to combine the first and third option of movement one can stop the local search procedure after a given number of iterations and then do path-relinking among the solution obtained in the local search and global best solution. Although combinations could be done, in the proposed algorithms, no combination of movements was implemented. Initial probabilities are assigned to each one of the three possible movements. These probabilities change as the algorithm runs. A general framework of the algorithm implemented in this work is listed in the following.

```
procedure PSO_TSP
{
  Define initial probabilities for particles' moves:
      p1 = x /*to follow its own way*/
      p2 = y /*to go forward pbest*/
      p3 = z /*to go forward gbest*/
      /* x+y+z=1 */
  Initialize a population of particles
  do
   for each particle p
    Evaluate particle p
    If the value of p is better than the value of pbest
       Then, update pbest with p
   end_for
   Define gbest as the particle with the best value
   for each particle p do
    Choose p's velocity
    Update p's position
   end_for
   Update probabilities:
      p1 = p1×0.95; p2 = p2×1.01; p3 = 100%-(p1+p2)
  while (a stop criterion is not satisfied)
}
```

At the beginning, the algorithm defines the probabilities associated with each velocity for the particles, where p_1, p_2 and p_3 correspond, respectively, to the likelihood that the particle follows its own way, goes toward the best previous position and goes toward the best solution found so far. Then, the algorithm proceeds modifying the particle's position according to the velocity operator randomly chosen. Finally, the probabilities are updated.

Initially, a high probability is set to p_1, and low values are assigned to p_2 and p_3. The goal is to allow that individualistic moves occur more frequently in the first iterations. During the execution this situation is being modified and, at the final iterations, p_3 has the highest value. The idea is to intensify the search in good regions of the search space in the final iterations.

Particles are initialized with a random adaptive version of the heuristic nearest neighbor [2]. The procedure is similar to the construction phase of a GRASP algorithm [7]. At first, a city is randomly chosen, then, other cities are added to the solution in each step. A restricted candidate list is built with the 5% cities closest to the last one inserted in the solution. A city is chosen at random from this list and is added to the solution. This step is repeated until a TSP solution is completed.

4 Computational Experiments

The proposed algorithms were implemented in C++ on a Pentium IV (3.0 GHz and 512 Mb of RAM) running Linux. The algorithms were applied to symmetric instances of the benchmark TSPLIB [22] with sizes ranging from 51 to 7397. The stop criteria were:

- To find the optimum
- To reach a maximum number of iterations = 2000
- To reach a maximum number of iterations with no improvement of the best current solution = 20
- To reach a maximum processing time = 60 seconds for instances with $n < 1000$, 300 seconds when $1000 \leq n < 5000$, 1000 seconds when $5000 \leq n < 7000$ and 2000 seconds for $n \geq 7000$.

The population had a fixed size with 20 particles.

A first experiment compared the two proposed algorithms in 11 symmetric instances with sizes ranging from 51 to 2103. Twenty independent runs of each algorithm were performed. The results are showed in Table 1 in terms of percent deviation from the optimal solution. This gap is computed with equation (3), where *Sol* and *Opt* denote, respectively, the (best or average) solution obtained by the algorithm and the optimal solution.

$$(Sol - Opt) \times 100/Opt \tag{3}$$

The columns show the name of the TSPLIB instances, the best solution (Min) and the average solution of the two versions of the proposed PSO algorithms denoted by PSO-INV and PSO-LK, the versions with the inversion and LK local search procedures, respectively.

Not surprisingly, PSO-LK exhibits a better performance than PSO-INV, since the local search procedure embedded in the former version is more powerful than the local search procedure of the latter version. However, a comparison of the results of the weakest version of the proposed algorithm with the ones reported in the work of Pang et al. [19] reveals that the former finds the best results. Once Wang et al. [26] reported results just for one asymmetric TSP instance, the authors implemented their algorithm in order to compare its performance with the proposed algorithm. Table 2 shows the best results (percent deviations) of the weakest version of the proposed algorithm, PSO-INV, and of the algorithms PSO-P [19] and PSO-W [26] for the three TSP symmetric instances with $n > 50$ of the computational experiment reported by Pang et al. [19].

Table 1. Comparison of the two versions of the proposed algorithms

Instances	PSO-INV		PSO-LK	
	Min	Av	Min	Av
eil51	0.2347	1.9836	0	0
berlin52	0	2.0041	0	0
eil76	2.4164	4.5167	0	0
rat195	5.8114	8.7581	0	0
pr299	5.8476	7.9952	0	0
pr439	4.4200	8.0111	0	0
d657	6.9656	9.6157	0	0
pr1002	9.8574	11.1900	0	0
d1291	13.2104	15.5505	0	0.0113
rl1304	10.4432	11.9942	0	0
d2103	16.7383	18.4180	0.0087	0.0267

Table 2. Comparison of two PSO heuristics with PSO-INV

Instance	PSO-W	PSO-P	PSO-INV
eil51	114	2.54	0.2347
berlin52	115	2.12	0
eil76	162	4.75	2.4164

The results showed in Table 2 correspond to the best solution found in 20 independent runs. The proposed algorithm PSO-INV was also applied to the asymmetric instance br17 reported by Wang et al. [26]. The results are showed in Table 3, where columns show the algorithm, the best and average percent deviation from the optimal solution, and the average runtime in seconds.

Table 3. Comparison of PSO-INV and PSO-W for the TSPLIB instance br17

Algorithm	Min	Average	T (s)
PSO-INV	0	0	< 0.01
PSO-W	0	16.66	60.04

A computational experiment investigated the differences between the results obtained by the LK procedure [2] and the PSO-LK algorithm. This experiment aimed at finding out if the proposed PSO approach was able to improve the LK [2] results. Table 4 shows the percent difference from the optimum, for 30 TSPLIB symmetric instances with n ranging from 439 to 7397. Bold elements mark when an improvement occurred. Twenty independent runs of each algorithm were performed. The columns T show the average time of each algorithm in seconds. From the thirty instances of the experiment, the PSO approach was able to improve twenty-three minimal results and all the averages. A statistical analysis shows that, the means of columns Min and Average of LK are 0.1115, 0.3431, respectively. The means of PSO-LK are 0.0022 and 0.0154. Although the runtimes of PSO-LK are higher than the LK, in average, improvements of 98% and 95% were achieved on the best and mean results, respectively.

Table 4. Comparison of LK and PSO-LK results

Instance	LK			PSO-LK		
	Min	Average	T (s)	Min	Average	T (s)
pr439	0.0000	0.0463	0.88	0.0000	**0.0000**	0.78
pcb442	0.0000	0.1119	0.67	0.0000	**0.0000**	0.80
d493	0.0029	0.1216	2.21	**0.0000**	**0.0000**	19.38
rat575	0.0295	0.1277	0.62	**0.0000**	**0.0000**	6.47
p654	0.0000	0.0078	3.72	0.0000	**0.0000**	1.90
d657	0.0020	0.1500	1.70	**0.0000**	**0.0000**	12.42
rat783	0.0000	0.0704	0.90	0.0000	**0.0000**	5.25
dsj1000	0.0731	0.2973	5.89	**0.0027**	**0.0031**	178.48
pr1002	0.0000	0.1318	1.96	0.0000	**0.0000**	9.50
u1060	0.0085	0.1786	3.12	**0.0000**	**0.0000**	38.18
vm1084	0.0017	0.0669	2.59	**0.0000**	**0.0010**	34.74
pcb1173	0.0000	0.1814	1.78	0.0000	**0.0001**	48.18
d1291	0.0039	0.4333	1.79	**0.0000**	**0.0000**	29.86
rl1304	0.0202	0.3984	2.67	**0.0000**	**0.0000**	21.62
rl1323	0.0463	0.2300	2.34	**0.0000**	**0.0092**	225.32
nrw1379	0.0547	0.1354	2.45	**0.0017**	**0.0085**	417.80
fl1400	0.0000	0.1215	14.87	0.0000	**0.0000**	15.42
fl1577	0.7371	2.2974	6.30	**0.0000**	**0.0135**	461.99
vm1748	0.0903	0.1311	4.66	**0.0000**	**0.0018**	854.17
u1817	0.1976	0.5938	1.88	**0.0000**	**0.0863**	789.18
rl1889	0.1836	0.3844	4.58	**0.0000**	**0.0073**	894.43
d2103	0.0597	0.3085	2.82	**0.0000**	**0.0043**	1137.56
u2152	0.2381	0.5548	2.16	**0.0000**	**0.0717**	1415.32
pr2392	0.0775	0.3904	3.78	**0.0000**	**0.0021**	577.78
pcb3038	0.1598	0.2568	5.42	**0.0101**	**0.0396**	323.94
fl3795	0.5665	1.0920	15.60	**0.0000**	**0.0142**	621.63
fnl4461	0.0882	0.1717	9.08	**0.0296**	**0.0462**	583.78
rl5915	0.3528	0.5343	10.08	**0.0122**	**0.0633**	1359.25
rl5934	0.2221	0.4761	10.65	**0.0012**	**0.0650**	983.04
pla7397	0.1278	0.2912	21.85	**0.0075**	**0.0253**	1563.22

Finally, Table 5 shows a comparison of the results obtained by PSO-LK and three effective heuristics: Tourmerge [4], NYYY iterated Lin-Kermighan variant (reported at http://www.research.att.com/~dsj/chtsp/) and JM iterated Lin-Kernighan variant. The results of these heuristics were obtained in the DIMACS Challenge page: http://www.research.att.com/~dsj/chtsp/results.html.

The columns related to PSO-LK show the best and average tours found in twenty independent runs. The columns related to the Tourmerge algorithm show the best and average tours obtained in five independent runs. Results are not reported for instances fnl446 and pla7397. The columns related to the NYYY and JM iterated Lin-Kernighan variants show the best tours obtained in ten n iterations runs.

Table 5. Comparison of heuristics for TSP symmetric instances

Instance	PSO-LK		Tourmerge		ILKNYYY	ILKJM
	Min	Average	Min	Average	Nb10	Nb10
dsj1000	0.0027	0.0031	0.0027	0.0478	0	0.0063
pr1002	0	0	0	0.0197	0	0.1482
u1060	0	0	0	0.0049	0.0085	0.0210
vm1084	0	0.0010	0	0.0013	0.0217	0.0217
pcb1173	0	0.0001	0	0.0018	0	0.0088
d1291	0	0	0	0.0492	0	0
rl1304	0	0	0	0.1150	0	0
rl1323	0	0.0092	0.01	0.0411	0.01	0
nrw1379	0.0017	0.0085	0	0.0071	0.0247	0.0018
fl1400	0	0	0	0	0	0
fl1577	0	0.0135	0	0.0225	0	0
vm1748	0	0.0018	0	0	0	0
u1817	0	0.0863	0.0332	0.0804	0.1643	0.2657
rl1889	0	0.0073	0.0082	0.0682	0.0082	0.0041
d2103	0	0.0043	0.0199	0.3170	0.0559	0
u2152	0	0.0717	0	0.0794	0	0.1743
pr2392	0	0.0021	0	0.0019	0.0050	0.1495
pcb3038	0.0101	0.0396	0.0036	0.0327	0.0247	0.1213
fl3795	0	0.0142	0	0.0556	0	0.0104
fnl4461	0.0296	0.0462	---	---	0.0449	0.1358
rl5915	0.0122	0.0633	0.0057	0.0237	0.0580	0.0168
rl5934	0.0012	0.0650	0.0023	0.0104	0.0115	0.1723
pla7397	0.0075	0.0253	---	---	0.0209	0.0497

From the twenty-one instances for which Tourmerge presented results, PSO-LK finds five best minimal results, Tourmerge finds three best minimal results and both algorithms find the same quality tours for thirteen instances. Regarding average results Tourmerge finds the best values on six instances and PSO-LK finds the best values on thirteen instances. The mean values of the "Min" and "Average" columns for the twenty-one instances are 0.0013 and 0.0186 for PSO-LK and 0.0041 and 0.0467 for Tourmerge. These results show that, in average, PSO-LK has a better performance than Tourmerge regarding minimal and average results.

Comparing the proposed algorithm with the NYYY iterated Lin-Kernighan variant, one can observe that from the twenty-three instances of the experiment, PSO-LK finds the best tours for thirteen instances, ILK-NYYY finds the best results for one instance and same results are found for nine instances. The averages of the best results of these two algorithms for the twenty-three instances are 0.0028 and 0.0199, PSO-LK and ILK-NYYY, respectively.

The comparison with the JM iterated Lin-Kernighan version shows that the PSO-LK algorithm finds the best results for sixteen instances and both algorithms find the same tours quality for seven instances. The averages for PSO-LK and ILK-JM are 0.0028 and 0.0569, respectively.

5 Conclusions

This paper presented a PSO algorithm for the TSP where the concept of velocity is differentiated when the particle has to follow its own way and when it goes forward someone's position. Local search procedures were applied in the former and path-relinking operations were applied in the latter case.

Although some works where PSO algorithms were applied to the TSP were presented previously, those algorithms did not reported results which could be compared with the ones obtained by effective heuristics. In this work, an effective PSO approach was presented for the TSP which results were compared with three high quality heuristics for this problem. The comparison showed that the proposed algorithm outperforms the three heuristics regarding best tours. The comparison among the average results of PSO-LK and the Tourmerge also shows that the first heuristics finds better results.

References

1. Aarts, E., Lenstra, J.K.: Local Search in Combinatorial Optimization. John Wiley & Sons, Chichester, England (1997)
2. Bellmore, M., Nemhauser, G.L.: The Traveling Salesman Problem: A Survey. Operations Research Vol. 16 (1968) 538–582
3. Concorde TSP Solver: http://www.tsp.gatech.edu/concorde.html. Last access on 01/18/2005
4. Cook, W.J., Seymour, P.: Tour Merging via Branch-decomposition. INFORMS Journal on Computing Vol. 15 (2003) 233-248
5. Dorigo, M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation vol. 1, N. 1 (1997) 53-66
6. Eberhart, R.C., Shi, Y.:Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation Vol. 1 (2000) 84-88
7. Feo, T.A., Resende, M.G.C.: A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. Operations Research Letters Vol. 8 (1989) 67–71
8. Glover, F.: Parametric Combinations of Local Job Shop Rules. Chapter IV, ONR Research Memorandum N. 117, GSIA, Carnegie Mellon University, Pittsburgh, PA (1963)
9. Glover, F., Laguna, M., Martí, R.: Fundamentals of Scatter Search and Path Relinking. Control and Cybernetics Vol. 29, N. 3 (2000) 653-684
10. Gutin, G., Punnen, A. P. (Ed.): Traveling Salesman Problem and Its Variations. Kluwer Academic Publishers (2002)
11. Heppner, F., Grenander, U.: A Stochastic Nonlinear Model for Coordinated Bird Flocks. In: Krasner, S. (eds.), The Ubiquity of Caos, AAAS Publications, Washington, DC (1990)
12. Holland, J. H.: Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI (1975)
13. Johnson, D. S., McGeoh, L.A.: Experimental Analysis of Heuristics for the STSP. In: Guttin, G., Punnen, A.P. (eds.): Traveling Salesman Problem and Its Variations, Kluwer Academic (2002)

14. Kennedy, J., Eberhart, R.: Particle Swarm Optimization, Proceedings of the IEEE International Conference on Neural Networks Vol. 4 (1995) 1942-1948
15. Lin, S., Kernighan, B.: An Effective Heuristic Algorithm for the Traveling-salesman Problem. Operations Research Vol. 21 (1973) 498-516
16. Machado, T.R., Lopes, H.S.: A Hybrid Particle Swarm Optimization Model for the Traveling Salesman Problem. In: Ribeiro, H., Albrecht, R.F., Dobnikar, A. (eds.): Natural Computing Algorithms, Wien: SpringerWienNewYork (2005) 255-258
17. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Caltech Concurrent Computation Program, C3P Report 826, 1989
18. Onwubulu, G.C., Clerc, M.: Optimal Path for Automated Drilling Operations by a New Heuristic Approach Using Particle Swarm Optimization. International Journal of Production Research Vol. 42, N. 3 (2004) 473-491.
19. Pang,W., Wang, K.-P., Zhou, C.-G., Dong, L.-J., Liu, M., Zhang, H.-Y., Wang, J.-Y.: Modified Particle Swarm Optimization Based on Space Transformation for Solving Traveling Salesman Problem. Proceedings of the Third International Conference on Machine Learning and Cybernetics (2004) 2342-2346
20. Pomeroy, P.: An Introduction to Particle Swarm Optimization. Electronic document available at www.adaptiveview.com/ipsop1.html
21. Reeves, W.T.: Particle Systems Technique for Modeling a Class of Fuzzy Objects. Computer Graphics Vol. 17, N. 3 (1983) 359-376
22. Reinelt, G.: TSPLIB, 1995. Available: http://www.iwr.uni-heidelberg.de/iwr/comopt/ software/TSPLIB95/
23. Reynolds, C. W.: Flocks, Herds and Schools: a Distributed Behavioral Model, Computer Graphics Vol. 21, N. 4 (1987) 24-34
24. Reynolds, R. G.: An Introduction to Cultural Algorithms. In: Proceedings of Evolutionary Programming, EP94, World Scientific, River Edge, NJ (1994) 131-139
25. Shi, Y., Eberhart, R.C.: Parameter Selection in Particle Swarm Optimization. Evolutionary Progamming VII, Proceedings of EP98, New York (1998) 591-600
26. Wang, K.-P., Huang, L., Zhou, C.-G., Pang, W.: Particle Swarm Optimization for Traveling Salesman Problem. Proceedings of the Second International Conference on Machine Learning and Cybernetics (2003) 1583-1585