Published in: Nedjah, N et al. (eds.) Proceedings of 5th International Conference on Hybrid Intelligent Systems, Rio de Janeiro, 06-09/november, IEEE Computer Society Press, p. 245-250, 2005

Particle Swarm Optimization with Fast Local Search for the Blind Travelling Salesman Problem

Heitor S. Lopes*
Bioinformatics Lab./CPGEI, CEFET-PR
Av. 7 de setembro, 3165
80230-901, Curitiba (PR), Brazil
hslopes@cpgei.cefetpr.br

Leandro S. Coelho
PPGEPS/LAS, PUC-PR
R. Imaculada Conceição, 1155
80215-901, Curitiba (PR), Brazil
leandro.coelho@pucpr.br

Abstract

The classical travelling salesman problem (TSP) is to determine a tour in a weighted graph (that is, a cycle that visits every vertex exactly once) such that the sum of the weights of the edges in this tour is minimal. Hybrid methods, based on nature inspired heuristics, have shown their ability to provide high quality solutions for the TSP. The success of a hybrid algorithm is due to its tradeoff between the exploration and exploitation abilities in search space. This work presents a new hybrid model, based on Particle Swarm Optimization and Fast Local Search, with concepts of Genetic Algorithms, for the blind TSP. A detailed description of the model is provided, emphasizing its hybrid features. The control parameters were carefully adjusted and the implemented system was tested with instances from 76 to 2103 cities. For instances up to 439 cities, the best results were less than 1% in excess of the known optima. In the average, for all instances, results are 2.538% in excess. Simulation results indicated that the proposed hybrid model performs robustly. These results encourage further research and improvement of the hybrid model to tackle with hard combinatorial problems.

1 Introduction

Many problems with important practical applications are concerned with the search of the "best" configuration or set of parameters to achieve some objective criteria. Such problems are generally referred to as optimization problems. If the entities to be optimized are discrete, the number of feasible solutions is finite. In this case, such problems are called combinatorial optimization problems. The travelling salesman problem (TSP), a classical example of a NP-hard problem, is one of the most widely discussed problems in combinatorial optimization. The TSP is the problem of finding the shortest closed tour (shortest Hamiltonian cycle) through a given set of n cities visiting each city exactly once and returning to the starting point, minimizing the total cost of the tour. The TSP and its variants have many practical applications such as X-ray crystallography, vehicle routing, integrated circuits design, automated guided vehicles scheduling, robot control, mobile computing, etc. In this work, we focus on the symmetric and blind TSP (BTSP). BTSP allows the starting point to be in any city and it is defined by a symmetric distance matrix between cities.

During the last decades, several algorithmic strategies have been proposed for finding near optimum solutions to the TSP, including 2-opt [4], particle swarm optimization [17], guided local search [16] and hybrid heuristics [2].

The computational demand of approximate and exact algorithms is huge. Some problem-independent and domain-specific heuristics have been proved to be very effective both in terms of execution and quality of the solutions achieved [2]. Domain-specific heuristics are very effective for refining arbitrary points in the search space into better solutions for solving TSPs. On the other hand, problem-independent heuristics perform quite poorly on large TSP instances [13]. They require more execution time for solutions whose quality is often not comparable with those achieved in much less time by their domain-specific local search counterparts.

Some of the most successful published results [7], [14], [13], have been provided by hybrid problem-independent heuristics that incorporate local optimization based on domain-specific heuristics. Local search techniques can often be incorporated naturally into problem-independent heuristics, such as evolutionary algorithms, tabu search or simulated annealing, in order to increase the effectiveness of the search. Such hybrid models have the potential to exploit the complementary advantages of problem-

*This work was partially supported by the Brazilian National Research Council – CNPq, under research grant no. 305720/04-0 to H.S. Lopes.

independent heuristics (robustness and exploration abilities), and problem-specific heuristics (fast convergence toward local minima and exploitation abilities).

This paper presents a new hybrid model for the TSP. The proposed model is based on the Particle Swarm Optimization – PSO [5],[6] heuristics, Fast Local Search – FLS [16], besides some concepts drawn from Genetic Algorithms – GA [8]. From PSO, the model uses local maximum, global maximum and the swarm movement. FLS is used to improve solutions found during the search, by evaluating points in the neighborhood of each particle. From GA, we used the representation of the solution as a numerical vector and the order crossover for moving particles across the discrete search space. We will show that the success of this new hybrid model is thanks to the tradeoff between the exploration and exploitation features of its composing paradigms.

2 Background

2.1 Particle Swarm Optimization

PSO is a heuristic method for optimization proposed by Eberhart and Kennedy [5],[6]. It is inspired in the behavior of social agents found in nature. This behavior can be observed in bird flocking, bee swarming, and fish schooling, for instance.

The computational model is population-based. Agents, or particles, change their position (state) in the multidimensional search space of the problem, according to own experience and the influence of the neighboring particles. Each particle has a limited store capability, keeping track only of information about its current position, speed and quality (fitness regarding the other particles), as well as its best position ever visited ("best local solution"– BLS). Amongst the swarm of particles, the one with best quality is referred as "the best global solution" – BGS. At each time tick, particles move, influenced by both BLS and BGS, to a new position in the search space. This is an iterative process, repeated until a stop condition is met, usually a predefined number of iterations. BGS is updated whenever a better solution than the previous is found. This procedure is similar to the principle of elitism, common in most GA applications, since throughout iterations the best solution is conserved. Although, there is a subtle difference: BGS is a reference for all particles in the same iteration (in GA, this would be similar to say that all individuals would mate with the best individuals). BLS is used only by a particle itself, not sharing this information with other particles.

It is interesting that BLS would be a point with good fitness but, also, it would be better if this point is quite far from BGS, so as to improve diversity. In population-based heuristics diversity maintenance throughout iterations is often a challenge, and it is a necessary condition to assure a satisfactory exploration of the search space. In PSO, when many BLS's are somewhat close to the BGS, there will be a particle crowding and the search stagnates. A mechanism to avoid the consequences of this unavoidable convergence will be described later.

In the classical PSO model, the movement of a particle is defined by Equation 1, where its next position (X_{i+1}) is updated using the current position and a speed term (V_i).

$$X_{i+1} = X_i + V_i \tag{1}$$

In fact, the speed term actually does not have the dimension of velocity. It could be better defined as ΔX_i but, for the sake of simplicity, it is called speed (V_i), following [6]. The speed term is defined according to the Equation 2:

$$V_i = c_1.r_1.d_{BLS} + c_2.r_2.d_{BGS} \tag{2}$$

where: V_i is the current speed of particle i; r_1 and r_2 are random values in the range [0..1]; c_1 and c_2 are the weights of BLS and BGS, respectively (in percentage); d_{BLS} and d_{BGS} are the distance between the current position and BLS and the current position and BGS, respectively. The speed term, that is, the updating rate of the current position, is directly proportional to the distance between the current position to BLS and BGS. Therefore, within few iterations the particle will be attracted to either BLS or BGS.

The speed term controls the amount of global and local exploration of the particle (that is, the balance between exploration and exploitation). A high speed facilitates global exploration, while small speed will encourage local search. A user-defined upper bound (V_{max}) is established to limit the maximum speed of particles. According to a psychological interpretation of PSO [6], the swarm of particles is like as a population of individuals. Then, the two terms of Equation 2 represent the cognitive and the social components of a particle's behavior. The former leads the particle to repeat its own past successful behaviors, while the latter makes it follows the others'. There are no default values for weights c_1 and c_2; sometimes they are set identical and sometimes they are set asymmetrical. It is commonly accepted that those weights are problem-dependent and this seems to be an open subject for further research [3].

As mentioned before, when particles agglomerate, a mechanism is necessary to avoid stagnation, and the crowd is dissolved by an explosion, repositioning all particles randomly in the search space. However, they do not loose the information of BLS and BGS. A similar approach applied to the classical PSO, named "mass extinction" was proposed by [18], where the population of particles becomes extinct at regular time intervals. The way particle agglomeration is treated in our model will be explained later.

2.2 Fast Local Search

The FLS algorithm [16] is an enhanced local search procedure, also known as fast hill climbing. Although FLS can achieve similar results than the classical hill-climbing (or neighborhood search), it is more efficient regarding the processing time. The main characteristic of FLS is the use of an activation bit for each position of the of the current solution vector. In the initial iteration, all bits of the binary activation vector are set to 1. Whenever two points are selected, the corresponding bits are reset. From the second iteration on, only the bits corresponding to permutations of higher fitness (than the previous one) are kept set. Therefore, the number of permutations is greatly reduced, since changes occur only for those positions where the corresponding bit is set. The intention of using FLS is to ignore points of the search space which are unlikely to lead to fruitful hill-climbs, by means of reducing the neighborhood considered for hill-climbing of a given point in the search space. The way actually FLS is used in our hybrid approach is detailed in section 3.3.

3 The hybrid PSO model

The hybrid model proposed in this work can be defined as a discrete PSO that uses explicit local search and concepts from GA. The original PSO was first devised to cope with continuous rather than discrete search space and, therefore, some adaptations were necessary. The hybrid nature of our model can be considered strongly coupled, since it embodies the fusion of concepts from different paradigms: the local search performed by FLS and the GA characteristic appropriated to deal with the permutation problem (OX operator [8]). In contrast, [12] present a loosely coupled model, where two different subsystems (GA and PSO) share only the final results. Other hybrid methods, with different approaches, can also be found elsewhere, such as [9] and [11], where either the components of the model interact in the same context, or the main paradigm uses concepts from other.

3.1 Particles, swarm and fitness

In our model, each particle represents a possible solution for the BTSP, that is, a complete tour. A given particle is composed by three main vectors, representing possible permutations for the BTSP: CP (current position), BLS and BGS. Also, the information of its current speed (V_i), current fitness of the particle ($C_{fitness}$) and fitness of BLS ($M_{fitness}$) are kept in the particle. Other global parameters are defined: maximum number of iterations (N_{max}), number of particles (Np), minimum distance for computing fitness (D_{min}), number of cities of the current problem (NC), and the matrix of distances between cities ($D = [d_{ij}]$). Parameter NC determines the size of vectors CP, BLS and BGS. As default, the distance matrix is generated using Euclidean coordinates in the plane.

Fitness is computed by dividing D_{min} by the cost of the tour represented by the current solution. Considering the benchmark instances used in this work, D_{min} is set to the already known optimal value of the tour. Therefore, fitness values represent the excess of distance relative to the known optimum for the instance. In the case when the value of optimum tour is not known, D_{min} can be set to 1. In this way, fitness is inversely proportional to the cost of the tour.

The swarm represents a population of particles and, at the startup, Np particles are generated (usually, $20 \leq Np \leq 50$). For each particle i, V_i is randomly initialized, respecting V_{max}, and vector CP is set with a random (but valid) tour. For the first iteration, BLS receives the value of CP. All just generated particles will be in different points of the search space, and all of them will have different values for CP. This represents a good diversity for the initial population, an essential feature for evolving good solutions. Next, the fitness of each particle is computed. The BGS and its fitness receives the corresponding values of the first generated particle and are updated as soon as any other particle is better than the previous stored value.

The speed term of Equation 1 requests two parameters: d_{BLS} and d_{BGS}. In the classical PSO, those parameters are continuous. Therefore, the distance between particles is calculated as the value of BLS or BGS decreased by the CP value. However, in the BTSP, each position of the search space represents a complete tour (a permutation of cities), thus requesting a new method to compute the distance between particles, inspired on the Hamming distance. Given two particles "A" and "B" representing a tour, the distance between them is computed comparing vectors departing from a common city (point zero). Initially, the distance is null and, for each position of the vector the corresponding values are compared. Whenever they are different, the distance is incremented by 1. In this way, the maximum distance possible will be exactly equal to NC. This computation may require a previous adjustment in one of the vectors: it will be slid circularly until the initial point of both vectors is the same. This procedure creates a new vector B′, but does not change the encoded information about the tour (recall that it refers to a BTSP).

3.2 Diversity and particles' movement

In each iteration, the average distance between particles and the BGS is calculated. If this value is lower than a given percentage of NC (recall how the distance between particles is computed), agglomeration is characterized and the swarm is exploded. This "hot boot" is controlled by a pa-

rameter named diversity (δ). Also, it is possible to have agglomeration around the BLS of a particle. In this particular case, all surrounding particles will have its BLS value changed to a random value. This procedure simulates a local explosion of the swarm.This diversity maintenance procedure does not affect the original number of iterations and it is essential for an efficient exploration of the search space, without excessive local exploitation.

The movement of particles is based on Equations 1 and 2 and the GA-inspired OX operator [8] that recombines two possible solutions. The OX operator is adapted as follows: only one cut point (P_1OX) is randomly chosen and it is the same for the two particles. The second cut point (P_2OX), necessary to define the matching section, is found traversing circularly $NC-1$ positions of the solution vector (tour represented by the particle). Operator OX is applied to two sets of vectors: {CP, BLS} and {CP, BGS}. The number of positions of the matching section for each operation is given by Equations 3 and 4, where c_1 and c_2 are the same parameters of Equation 2.

$$S_{BLS} = c_1.V_{id} \tag{3}$$

$$S_{BGS} = c_2.V_{id} \tag{4}$$

This procedure generates two new temporary solutions that are evaluated according to their fitness. The solution with best fitness will be considered the new CP of the particle. The concept of speed in this work is the same as in the classical PSO, and determines the level of exploration of the search space by the particles. As usual, after a particle has been moved in the search space, its BLS is updated accordingly. An iteration is defined by the movement of one particle and, after each iteration BGS is updated, if necessary.

3.3 Local search with FLS

Refinement of solutions takes place together with the search process. This refinement is accomplished by a local search, exploring positions (of the search space) surrounding to a given reference particle by means of the FLS strategy. For each particle, there is a binary activation vector. When two points of the activation vector are randomly selected, the changes in the solution vector (tour) is done using the 2-opt heuristics [15]. That is, the sub-tour defined by these two points is inverted leading to a different tour, but preserving its structure. This newly created solution is stored in a temporary memory. The remaining bits set of the activation vector are browsed two-by-two and the same procedure as above is repeated. As result, the temporary memory holds several variations descendent of the current solution. The best of them substitutes the current particle and the remaining are discarded. The 2-opt heuristics alone is not efficient [15], but its combined use with FLS in our hybrid model can enhance efficiency of the search, especially when the problem has many local maxima. A more detailed explanation about FLS is avoided due to space limitations.

4 Parameters adjustment

The hybrid PSO model proposed in this work has some parameters that control its behavior and performance. It is a matter of fact that, for the same problem instance, the behavior or the system can be very different and lead to diverse results, when using different sets of control parameters. Much work in the literature does not specify clearly how control parameters are adjusted, and does not give the attention this issue deserves. Aware of this, we conducted a series of experiments to investigate the effect of each control parameter in the performance of our system. Again, here, some simulation details, as well the sensitivity analysis, are not shown due to space limitations.

Each experiment was composed of 100 independent runs, using instances from the TSPLIB [1], a collection of instances widely used as a benchmark. For the adjustment of parameters, we used two BTSP instances: pr76 (with 76 cities), and pr299 (with 299 cities). These instances are not too large to be computationally expensive (for our system) and not too small to be trivial. The quality criterion was the distance of the Hamiltonian cycle found by the algorithm divided by the optimal known value, averaged over the 100 runs.

The initial values for the running parameters are shown in Table 1. Those values were based on the recent literature about PSO (except for FLS rate, which is specific for our system). The strategy used was to investigate the performance of the system changing some parameter(s) and selecting the two best values, and then using such values for investigating the next parameters.

The following parameters and ranges were tested: $c_1 = \{0.3, 0.4, 0.5, 0.6, 0.7\}$; $c_2 = \{0.3, 0.4, 0.5, 0.6, 0.7\}$; $V_{max} = \{0.5, 0.6, 0.7, 0.8, 0.9\}$ (this parameter corresponds to the proportion of NC); $\delta = \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40\}$ (defined in Section 3.2 as a proportion of NC); and FLS rate= $\{0.05, 0.10, 0.15, 0.20, 0.25\}$. After all experiments, the set of default parameters for our hybrid PSO was defined as those shown in Table 1.

5 Results

In this section, we report the application of our proposed system to several instances of the BTSP, also found

Table 1. Initial and default values for parameters of the hybrid PSO.

Parameter	Initial Value	Default Value
Number of particles	20	20
Number of iterations	1200	1200
C_1	0.4	0.7
C_2	0.6	0.3
δ	0.05	0.4
V_{max}	0.8	0.7
FLS rate	0.1	0.25

at TSPLIB. We tested two versions of our system: with (hybrid PSO) and without (pure PSO) the FLS feature. The experiments with FLS rate set to zero aimed at evaluating in what extent this distinguishing feature of our system influences the performance. Except by FLS rate, all other parameters were set to the default values previously defined. Besides the instances used to investigate the best set of control parameters (pr76 and pr299), the following instances of TSPLIB were used: rat195, pr439, d657, pr1002, rl1304, d1655 and d2103. The suffix of each instance is the number of cities it represents, that is, from 76 to 2103 cities. It is important to observe that most work in the literature regarding heuristic methods for the TSP use small instances.

Our hybrid PSO system was implemented in ANSI C programming language and all tests reported here were run in a desktop computer. Results are presented in Table 2. In this table, for each instance, we present the best solution found and the average over 10 independent runs. Values shown are normalized with respect the known optimal tour for each instance, and represent the percent of excess over it. Therefore, the small the excess, the better and closer to the optimal value. Considering all nine instances reported, the average excess was 2.538% for the hybrid PSO and 87.006% for the pure PSO.

Table 2. Results obtained.

	hybrid PSO		pure PSO	
Instance	Best	Avg.	Best	Avg.
pr76	0.000	0.000	47.581	53.107
rat195	0.810	1.148	75.696	76.757
pr299	0.120	0.620	85.878	86.535
pr439	0.280	0.500	88.408	88.962
d657	2.114	3.193	89.743	90.160
pr1002	3.569	4.715	93.108	93.270
rl1304	1.454	2.498	95.774	95.793
d1655	4.887	5.644	93.232	93.683
d2103	3.433	4.524	96.338	96.362

In PSO, an iteration corresponds to the evaluations of the whole swarm. The upper bound for the computational effort to solve a problem is obtained by the product of the number of particles by the number of iterations (in our case, 20 x 1200 = 24000). The real computational effort can be estimated by the product of the actual iteration in which the best solution was found by the number of particles, giving the number of evaluations. This measure is computer-independent. Table 3 shows, for each instance, the average iteration in which the best solution was found, as well the corresponding computational effort.

Table 3. Computational effort.

	hybrid PSO		pure PSO	
Instance	Avg.Iter.	Effort	Avg.Iter.	Effort
pr76	164	3280	1160	23200
rat195	853	17060	1156	23120
pr299	1010	20200	1173	23460
pr439	1061	21220	1163	23260
d657	986	19720	1183	23660
pr1002	1064	21280	1180	23600
rl1304	1071	21420	1184	23680
d1655	1109	22180	1167	23340
d2103	977	19540	1188	23760
average	922	18433	1173	23453

6 Conclusions

The dichotomy between a global search and a local search is a recurring theme in computational models of evolution and biology. In computational contexts, the hybridization of global and local search is known to produce more efficient optimization algorithms for the BTSP. Several general-purpose heuristics have been proposed for the BSTP. Conventional GA and PSO suffer from a certain inefficiency, characterized by a slow convergence and a lack of accuracy when a high-quality solution is required, especially for large instances of BSTP. In contrast, specialized, local search methods can focus in a given region of the search space, possibly retrieving the best local optima of the neighborhood. The aim of this paper was to propose a new hybrid model, combining PSO, FLS and GA methodologies for the BSTP. The combined features of PSO and GA were competent to find high-quality tours which are then refined by means of the FLS heuristic. While the PSO is efficient for searching the solution space globally (exploration), FLS does a good job while searching locally (exploitation).

The new hybrid model presented very good computational results, considering the performance measured with instances form 76 to 2013 cities of BSTP. The local search with FLS has a very significant influence on the final performance of hybrid model. Table 2 supports this assertion by showing how bad are results given by the version of PSO without FLS (pure PSO). Although not shown here,

the analysis of the joint effect of FLS and swarm explosion showed that they are responsible for keeping a high level of diversity along the search.

As expected, the quality of solutions found by the proposed model decreases as the size of the problem increases. For instances up to 439 cities, the best results found by our hybrid PSO were less than 1% in excess of the optimal value. A statistical analysis of such values and the proximity of the average with the best values suggest that the proposed system offers consistent results throughout different runs. Computational requirements with the proposed hybrid model are reduced compared to the pure PSO, as shown in Table 3. The pure PSO obtained very poor results for all instances, even requiring a higher computational effort than the hybrid PSO. This fact confirms the need for hybridism as a way to better explore the search space.

Regarding the processing time, using the control parameters previously defined, the hybrid PSO took around six times more time to run than the pure PSO. For real-world applications this fact can be neglected considering the quality of the solutions found by the hybrid PSO compared with the pure PSO.

There are many approaches devised to solve TSP, including those that use PSO [17],[12]. Usually, such approaches are tested with small dimension problems. For instance, [17] reports the application their model to instances up to only 14 cities. Real-world problems usually have a much larger dimensionality, for which more efficient models must be used, such as the one here proposed.

Hybrid heuristic models are interesting for hard problems since they combine good quality features from several techniques in a single paradigm. The concepts of GA and FLS embodied in the PSO paradigm have lead to a robust and efficient model, therefore justifying the need for hybridism. Results can be considered very good for an heuristic method, in comparison with other similar methods in the recent literature. It is worth to emphasize the use of FLS with 2-opt. This strategy has enhanced solutions found by the algorithm, but at the expense of a larger, but acceptable, computational cost. Our results confirm the fact that most of successful heuristic approaches for the TSP do rely on hybrid strategies that include some kind of local search.

Results encourage further work that will comprise the study of a less expensive FLS, as well as other strategies that could improve the model, such self-adaptive parameters [10], GA's concepts of niches and species [8] and breeding and subpopulations [9].

References

[1] *http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.*

[2] R. Baraglia, J. I. Hidalgo, and R. Perego. A hybrid heuristic for the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 5(6):613–622, 2001.

[3] M. Clerc. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, pages 1951–1957, 1999.

[4] G. Croes. A method for solving traveling salesman problems. *Operations Research*, 6(6):791–812, 1958.

[5] R. C. Eberhart and J. Kennedy. Particle swarm optimization. In *Proc. IEEE Int. Conf. on Neural Networks*, pages 1942–1948, 1995.

[6] R. C. Eberhart and J. Kennedy. *Swarm Intelligence*. Morgan Kaufmannr, San Francisco, CA, 2001.

[7] B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proc. IEEE Int. Conf. on Evolutionary Computation*, pages 616–621, 1996.

[8] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[9] M. Lvbjerg, T. K. Rasmussen, and T. Krink. Hybrid particle swarm optimizer with breeding and subpopulations. In *Proceedings of Genetic and Evolutionary Computation Conference*, volume 1, pages 469–476, 2001.

[10] M. Maruo, H. Lopes, and M. Delgado. Self-adapting evolutionary parameters: Encoding aspects for combinatorial optimization problems. *Lecture Notes in Computer Science*, 3448:154–165, 2005.

[11] S. Naka, T. Genji, T. Yura, and Y. Fukuyama. A hybrid particle swarm optimization for distributionstate estimation. *IEEE Transactions on Power Systems*, 18(1):60–68, 2003.

[12] X. H. Shi, L. M. Wan, H. P. Lee, X. W. Yang, L. M. Wang, and Y. C. Liang. An improved genetic algorithm with variable population size and a pso-ga based hybrid evolutionary algorithm. In *Proc. 2^{nd} IEEE Int.Conf. on Machine Learning and Cybernetics*, volume 3, pages 1735–1740, 2003.

[13] H.-K. Tsai, J.-M. Yang, Y. F. Tsai, and C.-Y. Kao. An evolutionary algorithm for large traveling salesman problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(4):1718–1729, 2004.

[14] S. Tsubakitani and J. R. Evans. An empirical study of a new metaheuristic for the traveling salesman problem. *European Journal of Operational Research*, 104(1):113–128, 1998.

[15] M. G. A. Verhoeven, E. H. L. Aarts, and P. C. J. Swinkels. A parallel 2-opt algorithm for the traveling salesman problem. *Future Generation Computer Systems*, 11(2):175–182, 1995.

[16] C. Voudouris and E. Tsang. Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469–499, 1999.

[17] K. P. Wang, L. Huang, C. G. Zhou, and W. Pang. Particle swarm optimization for the traveling salesman problem. In *Proc. 2^{nd} IEEE Int. Conf. on Machine Learning and Cybernetics*, volume 3, pages 1583–1585, 2003.

[18] X.-F. Xie, W.-J. Zhang, and Z.-L. Yang. Hybrid particle swarm optimizer with mass extinction. In *Proc. Int.Conf. on Communications, Circuits and Systems*, volume 2, pages 1170–1173, 2002.