In: B.Ribeiro et al., Adaptive and Natural Computing Algorithms. Wien: SpringerWienNewYork, pp. 255-258, 2005.

A hybrid particle swarm optimization model for the traveling salesman problem

Thiago Rogalsky Machado, Heitor Silvério Lopes *

Laboratório de Bioinformática /CPGEI, CEFET-PR
Av. 7 de setembro, 3165 – 80230-901 Curitiba (PR), Brasil
e-mails: {tmachado, hslopes}@cpgei.cefetpr.br

Abstract

This work presents a new hybrid model, based on Particle Swarm Optimization, Genetic Algorithms and Fast Local Search, for the blind traveling salesman problem. A detailed description of the model is provided. The implemented system was tested with instances from 76 to 2103 cities. For instances up to 299 cities, results are less than 1% in excess of the known optima. In the average, for all instances, results are 3.8317 % in excess. These excellent results encourage further research and improvement in the hybrid model

1 Introduction

This paper presents a new hybrid model for the traveling salesman problem (TSP). The proposed model is based on the Particle Swarm Optimization (PSO) [1,2] heuristics and uses concepts of Genetic Algorithms (GA) [3] and Fast Local Search (FLS) [4]. From PSO, the model uses local maximum, global maximum and swarm movement. 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. FLS is used to improve solutions found during the search, by evaluating points close to each particle.

2 Particle Swarm Optimization

PSO is a heuristic method for optimization proposed by Eberhart and Kennedy [1,2], and is inspired in the behavior of social agents. In nature, this behavior can be observed in bird flocking, bee swarming, and fish shooling, for instance.

The computational model is population-based, where agents, called 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 the principle of elitism, common to 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 it would be better if this point is far enough from the BGS to improve diversity. In population-based heuristics diversity maintenance throughout iterations is often a challenge, but 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).

$$X_{i+1} = X_i + V_i \quad (1)$$

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) [2]. The speed term is defined according to Equation 2:

$$V_i = c_1.r_1.dBLS + c_2.r_2.dBGS \quad (2)$$

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 weight of BLS and BGS, respectively (in percentage); *dBLS* and *dBGS* 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

* Corresponding author.

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 (*Vmax*) is established to limit the maximum speed of particles.

As mentioned before, when particles agglutinate, 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 lose the information of BLS and BGS. A variation of the classic PSO uses "mass extinction" [5], where the population of particles becomes extinct at regular time intervals. In our model, particle agglutination is treated in a similar way, and will be explained later.

3 Genetic Algorithms

A Genetic Algorithm (GA) is a search and optimization heuristic based on the Darwinian principle of the species evolution, where individuals better fitted to the environment are able to survive longer and propagate their genetic material to more descendants. A GA is also a population-based method with individuals that represent a possible solution for the problem. The genetic load is usually represented by a string of elements (genes). Individuals of a population are chosen according to their quality (fitness) to reproduce. Reproduction takes place by means of the application of genetic operators (crossover and mutation, for instance) to the selected pool of individuals, creating a new population. In particular, crossover is a genetic operator responsible for fostering local search, recombining pieces of two (or more) individuals. For permutation problems in combinatorial optimization, many crossover operators were proposed (see [3], for instance). In this work we used a strategy inspired on the Order crossover (OX). This operator works as follows: given two individuals, their genetic material is aligned and two random cut points are selected. The region of the chromosome between these two cut points is called matching section and will be exchanged between individuals. The remaining genes are mapped according to the matching section and are submitted to a sliding motion, so as to fill up the entire chromosome. In section 6 it will be explained how OX is used in our hybrid model.

4 Fast Local Search (FLS)

The FLS algorithm [4] an enhanced local search procedure, also known as fast hill climbing. Although FLS can achieve similar results than the classic hill climbing (or neighborhood search), it is more efficient, concerning processing time. The main characteristic of FLS is the use of an activation bit for each position of the of the current solution vector. At 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. This way, the number of permutations is greatly reduced, since changes occur only for those positions where the corresponding bit is set.

5 Traveling Salesman Problem

The TSP is a classical problem of combinatorial optimization and its modeling is of great interest for Computation and Engineering. For solving TSP, many methods have been proposed, including heuristic ones [3,4,6,7,8]. The simplest TSP considers a set of interconnected cities with symmetric distances between two points. The problem is to find the shortest path for visiting all cities passing only once at each point and returning to the initial city. There are many other variations of the problem, such as asymmetric distances between cities, capacitated vehicles, fuel/depot points with mandatory passing, time-windows restrictions, and so on. The blind TSP (BTSP) allows the starting point to be in any city and it is defined by a symmetric distance matrix between cities $\boldsymbol{D}=[d_{ij}]$ which gives the distance between cities I and j. A tour (t) can be represented as a cyclic permutation [4]. Let j be the next city visited after city i, in tour t. The cost of a tour, defined in Equation 3, represents the total distance traveled, where NC is the number of cities.

$$g(t) = \sum_{i=1}^{NC} d_{ij} \tag{3}$$

In the TSP library (TSPLIB), at the Internet[1], it is available a large collection of instances previously used as a benchmark. Those instances are used for evaluating performance of computational methods, since the optimal tours are always known.

6 Methodology

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 devised to cope with a continuous rather than discrete search space. Therefore, some adaptations were necessary. The hybridism of our model can be considered strongly coupled, since the GA characteristic appropriated to deal with the permutation problem (operator OX) embodied in the model. In contrast, Shi et al. [9] present a loosely coupled model, where two different subsystems (GA and PSO) share only the final results. The fusion of concepts from different paradigms in our work can also be found elsewhere, such as [10], where all components of the model interact in the same context, with mutual dependency.

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

6.1 Initial swarm and particles

In our model, each particle represents a possible solution for the BTSP (a complete tour). A given particle is composed by the following information: current position (CP), BLS, current speed (V_i), current fitness of the particle (*Cfitness*) and fitness of BLS (*Mfitness*). Besides the particle-related parameters, 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 problem (*NC*), and matrix of distances between cities ($\boldsymbol{D}=[d_{ij}]$). Parameter *NC* determines the size of vectors used in CP, BLS and BGS. The distance matrix is generated using Euclidean coordinates in the plane as default.

Fitness is computed by dividing D_{min} by the cost of the tour represented by the current solution,. Considering the instances used in this work, D_{min} is set to the optimal value of the tour and, therefore, fitness values represent the excess of distance relative to the known optimum for the instance. When the value of optimum tour is not known, D_{min} can be set to 1. Therefore, fitness is always inversely proportional to the cost of the tour, as in a minimization problem.

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 *Vmax* and the 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. 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: *dBLS* and *dBGS*. In the classic 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 vector 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 last procedure creates a new vector B', but does not changes the encoded information about the tour (recall that it refers to a BTSP).

6.2 Diversity and movement

In every iteration, the average distance between particles and the BGS is calculated. If this value is 5% lower than *NC* (recall how the distance between particles is computed), agglutination is characterized and the swarm is exploded, as mentioned before. Also, it is possible to have agglutination 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 is essential for an efficient exploration of the search space.

The movement of particles is based on Equations 1 and 2 and the OX operator 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 *S-1* positions of the solution vector (tour represented by the particle). Operator OX is applied to two sets of vectors: CP and BLS, and CP and BGS, and the number of positions of the matching section for each operation given by Equations 4 and 5, where c_1 and c_2 are the same parameters of Equation 2.

$$S_{MSL} = c_1 * V_{id} \tag{4}$$

$$S_{MSG} = c_2 * V_{id} \tag{5}$$

This procedure generates 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.

6.3 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. When two points of the activation vector are selected the changes in the solution vector (tour) is done using the 2-opt heuristics [7]. 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 is discarded.
The 2-opt heuristics alone is not efficient [7], but its combined use with FLS in our hybrid model can enhance efficiency of the search, especially when the problem has many local maxima.

7 Results

Preliminary tests were done exhaustively to adjust the parameters of the model to maximum performance, considering accuracy and processing time. These tests used two instances (pr76 and pr299), and the parameters shown in Table 1 were those that produced the best results. In that table, the parameter "FLS rate" is the probability of using local search in a given iteration. Further tests were done using the following instances of the blind TSP found in the TSPLIB: pr76, rat195, pr299, pr439, d657, pr1002, d1291 and d2103, for which the number cities ranges from 76 to 2103.
The model was implemented in ANSI C programming language and tests were run in a PC-clone computer with Athlon XP 2.4 processor and 512 Mbytes of RAM. Results are presented in Table 2, where "Excess" represent how far is the solution from the known optimal value. Each value of this column is, in fact, the average of 100 independent runs. Considering all nine instances, the average excess was 3.8317%.

Table 1: Initialization parameters for all problems.

Parameter	Value
Number of particles	20
Number of iterations	1200
C_1	0.7
C_2	0.3
Vmax	70% of the # of cities
FLS rate	0.15

Table 2: Results obtained for several instances.

Problem	# of cities	(%) Excess
pr76	76	0.0000
rat195	195	0.9834
pr299	299	0.5900
pr439	439	2.9561
d657	657	3.8490
pr1002	1002	6.6992
d1291	1291	4.5811
rl1304	1304	3.2456
d2103	2103	7.7493

8 Discussion and conclusions

As expected, the quality of solutions found by the proposed model decreases as the size of the problem increases. For instances up to 299 cities, we obtained average results less than 1% of excess, and the model was able to find the optimum at least once (recall that results shown is the average of 100 runs).
There are many models devised to solve TSP, including those that use PSO [8,9]. For instance, Wang, et al. [9] relates the application their model small instances (up to 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 excellent for an heuristic method, when compared 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.
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 as GA's concepts of niches and species [3] and breeding and subpopulations [10].

References

[1] Eberhart R.C., Kennedy J. (1995) Particle swarm optimization. In: Proc. IEEE Int. Conf. on Neural Networks, vol. 4, pp. 1942-1948

[2] Eberhart R.C., Kennedy J. (2001) Swarm Intelligence, Morgan Kaufman, San Francisco

[3] Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization & Machine Learning, AddisonWesley, Reading

[4] Voudouris C., Tsang E. (1999) Guide local search and its application to the traveling salesman problem. European Journal of Operational Research 113: 469-499

[5] Xie, X., Zhang, W., Yang, Z. (2002) Hybrid particle swarm optimizer with mass extinction. In: Proc. Int. Conf. on Comm., Circuits and Systems, pp. 1170-1173

[6] Pepper, J.W., Golden, B.L. (2002) Solving the traveling salesman problem with annealing-based heuristics: a computational study. IEEE Transactions on System, Man, and Cybernetics, Part A: Systems and Humans 32: 72-77

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

[8] Wang, K.P., Huang, L., Zhou, C.G., Pang, W. (2003) Particle swarm optimization for traveling salesman problem. In: Proc. 2nd IEEE Int. Conf. on Machine Learning and Cybernetics, pp. 1583-1585

[9] Shi, X.H., Wan, L.M., Lee, H.P., Yang, X.W., Wang, L. M., Liang, Y.C. (2003) An improved genetic algorithm with variable population size and a PSO-GA based hybrid evolutionary algorithm. In: Proc. of 2nd IEEE Int. Conf. on Machine Learning and Cybernetics, pp. 1735-1740

[10] Lvbjerg, M., Rasmussen, T. K., Krink T. (2001) Hybrid particle swarm optimizer with breeding and subpopulations. In: Proc. of Genetic and Evolutionary Computation Conference, pp.469-476