A Hybrid Particle Swarm Optimization – Simulated Annealing Algorithm for the Probabilistic Travelling Salesman Problem

Guillermo Cabrera G. [1], Silvana Roncagliolo D. [1], Juan P. Riquelme[1], Claudio Cubillos[1], Ricardo Soto [1,2]

[1] Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Av. Brasil 2241, Chile.

[2] Universidad Autónoma de Chile, Pedro de Valdivia 641, Santiago, Chile

{guillermo.cabrera, silvana, claudio.cubillos, ricardo.soto}@ucv.cl, juan.riquelme@mail.ucv.cl

Abstract: The Probabilistic Traveling Salesman Problem (PTSP) is a variation of the well known Traveling Salesman Problem (TSP). This problem arises when the information about customers demand is not available at the moment of the tour generation and/or the tour re-calculating cost is too elevated. In this article, a Hybrid Algorithm combining Particle Swarm Optimization (PSO) and Simulated Annealing (SA) is proposed, in order to solve the PTSP. The PSO heuristic offers a simple structured algorithm which supplies a high level of exploration and fast convergence, compared with other evolutionary algorithms. The SA algorithm is used to improve the particle diversity and to avoid the algorithm being trapped into local optimum. Two well-known benchmarks of the literature are used and the proposed PSO-SA algorithm obtains acceptable results. In fact, the hybrid algorithm improves the performance of simple PSO algorithm for all instances.

Keywords: Hybrid Algorithm, Metaheuristics, Routing Problems, Stochastic Optimization, Swarm Intelligence.

1. Introduction and Literature Review

The Travelling Salesman Problem (TSP) has several different variations. One of them is the stochastic variant called Probabilistic Traveling Salesman Problem (PTSP), which can be classified as a Stochastic Combinatorial Optimization Problem (SCOP). PTSP was proposed by Jaillet [19] and several authors have solved this model using different approaches. In [19] the PTSP is described as follows: Consider a set of n points which must be visited with probability *p*. On any given instance of the problem only a subset consisting of *k* out of the n points (0<*k*<n) have to be visited. The *k* number is determined according to a known probability distribution (such as the binomial). An *a priori* tour through all n points has to be found. Each point must be included only once in the *a priori* tour. On any given instance of the problem, the present k points will then be visited in the same order as they appear in the *a priori* tour. The PTSP is usually tackled by an *a priori* optimization phase [4] which comprises two stages: First, as said above, an *a priori* solution is found. In the second stage, the *a posteriori* solution is derived from the *a priori* solution. For *a posteriori* solution, the edges are visited in the same order as in the *a priori* solution, but excluding the edges that do not require to be visited. Fig. 1 shows both different *a priori* and *a posteriori* tours. In the example only odd edges are visited in the *a posteriori* tour. The problem of finding such an *a priori* tour which is of minimum length in the expected value sense is defined as a PTSP.

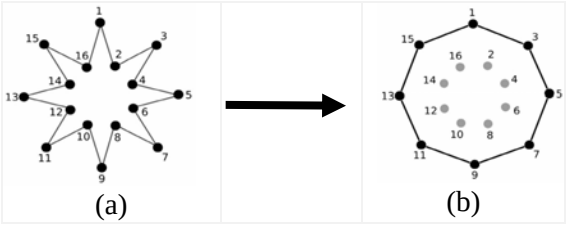

Figure 1. (a) *A priori* tour for 16 edges. (b) *A posteriori* tour after a realization.

The problem appears when the information about customers demand is not available at the moment of tour generation and/or the tour re-calculating cost is too elevated. In [19] the proposed model was solved by a well-known "hill climbing" algorithm. Thereafter, several authors have used different stochastic approaches in order to solve this optimization problem, e.g., in [5, 6] the authors solved the homogeneous PTSP using an Ant Colony Optimization (ACO) approach. In there, a variation of traditional ACO algorithm, called probabilistic Ant Colony System (pACS), raised very good solution for the PTSP, being as one of the more efficient strategies for stochastic routing problems. The most important difference between pACS and ACS is the set of arcs on which the pheromone is globally increased. One of the main conclusions of these works is that the pACS

algorithm is very competitive when the probabilities of the cities are far from 1. In this case (probabilities close to 1) the well-known ACS algorithm raised better solutions than pACS. In [26] the Expanding Neighborhood Search (ENS) is proposed for the resolution of the PTSP. The ENS is a variant of the well-known Greedy Randomized Adaptive Search Procedure (GRASP) which obtained a very good solution for a theoretical dataset. In [25] the author proposed a set of initial solution generators under a genetic algorithm (GA) framework for solving the PTSP. These three initial solution generators allow the GA to reduce computational time without decreasing the quality of the best reached solution. In [2, 3], the authors present an Estimation-Based algorithm involving local search strategies. In both, the authors experiment with different metaheuristics applying an estimation-based customization strategy, to evaluate the solution cost of different PTSP instances, obtaining excellent results. In [27] an hybrid algorithm - nature inspired - based on PSO, GRASP and ENS strategy is proposed, for a solution of the PTSP. The GRASP and ENS strategies are used in order to improve the quality of initial solutions for the PSO algorithm, which have as main characteristic the use of multiple swarms, reducing considerably the required computational time.

Finally, in [1] the authors describe a variant of particle swarm, called hybrid swarms, that incorporates an explicit selection mechanism similar to that used in more traditional evolutionary computations. The main author's conclusion is that the addition of selection supplies some advantage to particle swarm on certain functions.

The main contribution of this article is the application of an hybrid technique on a SCOP, using a simple but effective sort algorithm which improves considerably the solution reached by the hybrid algorithm in a reasonable computational time.

2. Mathematical Formulation

The PTSP, as mentioned above, was introduced by Jaillet [19]. The following mathematical formulation is extracted from [4, 5, 6]: In the PTSP, it is unknown in advance whether a node requires to be visited, but its probability of requiring a visit is given. The most widely used approach to tackle the PTSP is to construct an *a priori* solution before knowing which nodes require to be visited. Let $N=\{i/i=1,2,.....,n\}$ be a set of n customers. For each pair of customers $(i,j)\in N$, $d(i,j)$ represents the distance between i and j. In this case, it is assumed that the distances are symmetric, that is, $d(i,j)=d(j,i)$. An *a priori* tour $\lambda=(\lambda(1),\lambda(2),.....,\lambda(n))$ is a permutation over N, that is, a tour visiting all customers exactly once. Given the independent probability p_i that customer i requires a visit, q_i=1-p_i is the probability that i does not require a visit. Once the set of nodes that require to be visited is known, the *a posteriori* solution is derived by visiting the nodes that require to be visited in the order prescribed by the *a priori* solution and by skipping the nodes that do not require to be visited. As mentioned in section 1, the objective of the PTSP is to find an a priori solution, such that the expected cost of its associated *a posteriori* solution is minimized.

The general case where customers probabilities p_i may be different, is referred to as heterogeneous PTSP, however, when all probabilities are equivalent ($p_i = p$ for every customer *i*) the problem is called homogeneous PTSP. This article focuses on the homogeneous PTSP. The following convention for any customer index *i* is used:

$$i\begin{cases} i \bmod n & iff \quad i\neq 0 \ and \ i\neq n \\ n & otherwise, \end{cases} \tag{1}$$

The reason for defining the above convention is that numbers from 1 to n (and not from 0 to n − 1) were wanted to used as customer index and it was needed to make computations with the *mod* operator.

The expected length of an *a priori* tour λ can be computed in $O(n^2)$ time with the following expression derived by Jaillet [19]:

$$E[L(\tau)]=\sum_{i=1}^{n}\sum_{r=1}^{n-1} d(\xi)\, p_{\lambda(i)} p_{\lambda(i+r)} \prod_{i+1}^{i+r-1} q_{\tau}$$
$$with \tag{2}$$
$$d(\xi)=d(\lambda(i)+\lambda(i+r))$$

The following notation is used for any $i, j \in \{1,2,.....,n\}$

$$\prod_{i}^{j} q_\tau \begin{cases} \prod_{t=i}^{j} q_\tau(t) & \text{iff } 0 \le \text{j-i<n-1} \\ \prod_{t=i}^{n} q_\tau(t) \prod_{u=1}^{j} q_\tau(u) & \text{iff i-j>1} \\ 1 & \text{otherwise.} \end{cases} \quad (3)$$

The expression for the objective function (2) has the following intuitive explanation: each term in the summation represents the distance between the i^{th} customer and the $(i+r)^{th}$ customer, weighted by the probability that the two customers require a visit $p_{\lambda(i)} p_{\lambda(i+r)}$, while the r − 1 customer between them does not require a visit $\left(\prod_{i+1}^{i+r-1} q_\tau\right)$.

In the homogeneous PTSP, where $p_i = p$ and $q_i = q$ for every customer i, the expression for the objective function still requires $O(n^2)$ computation time, but it is a bit simpler than Eq. (2):

$$E[L(\tau)] = \sum_{i=1}^{n} \sum_{r=1}^{n-1} p^2 q^{r-1} d\left(\lambda(i) + \lambda(i+r)\right) \quad (4)$$

3. Hybrid Algorithm

Particle Swarm Optimization and Simulated Annealing Algorithms.

The PSO algorithm was proposed and developed by Kennedy and Eberhart [10, 11, 20, 21, 22]. It is considered as a more-robust methodology in the context of Swarm Intelligence (SI), and has proven to be an excellent alternative in order to solve discontinuous combinatorial optimization problems (COP). In fact, several articles have used this technique to solve complex COPs, particularly the classical TSP and its extension PTSP. Example of these applications on both TSP and PTSP are [27, 33, 35] among others. In [27], a new hybrid algorithmic nature inspired approach based on Particle Swarm Optimization (PSO), Greedy Randomized Adaptive Search Procedure (GRASP) and Expanding Neighbourhood Search (ENS) Strategy is proposed for the solution of the PTSP.

The main difference of this algorithm from the ones that uses more than one swarm is a feedback procedure that is used to, initially distribute the information (i.e., the good solutions of each swarm) in all other swarms and, afterwards, to help all the particles of all swarms to follow this information in order to finally find a new better solution. However, this multi-swarm approach requires a considerable computational time for its execution. For this reason, the authors hybridize the algorithm with two different procedures, one as a speeding up technique (the ENS) and one for the production of good initial solutions (the ENS-GRASP). In [33], two PSO-based algorithms are presented. The first one to solve the classic TSP and the other one to solve the generalized TSP. In both, the main characteristic is the use of an uncertain searching strategy and a crossover eliminated technique to accelerate the convergence speed. The use of this strategy allows the algorithm to solve problems with more cities compared with the SI existing algorithms for solving TSP. In [35], a PSO algorithm is applied to the TSP. The main characteristic of this implementation is the improvement produced by the introduction of both, the information communication strategy among the particles and the dynamic work allocation among the classes of particles swarm defined by the author. The first strategy is based on the Greddy's idea in order to strengthen the diversity of the particles and to speed the convergence process. The dynamic work allocation is implemented in addition to the first strategy, in order to promote the searching efficiency and solution quality. The main characteristic of this strategy is the definition of a sub-set of particle which have different searching strategies and tasks.

Other recent application of PSO can be found in [13, 24, 32, 36]. In [24] a time varying PSO is used in order to solve a problem of the industrial drives. This approach simplifies the fuzzy logic controller tuning procedure (used for finding feasible solutions), that reduces the amount of time needed for the problem resolution and provides good system performance. In [32], a simple PSO algorithm is applied to the Neutron Images Restoration problem. The solutions raised by the PSO algorithm were compared with other mathematics-based techniques. The experiments showed that the PSO algorithm reached excellent results and good efficiency in noisy image restoration. In [13] the authors proposed an original PSO algorithm which consists in using the concept of proportional

likelihood with modifications, a technique that is used in data mining applications, instead of the standard vector of velocities. In [36] the authors applied a PSO algorithm on a flow-shop problem, obtaining very competitive results compared to GA implementations from the literature.

Nevertheless, the PSO algorithm may be trapped into local optima, if the global best and local best positions are equal to the position of the particle over a number of iterations [29, 34]. Unlike other evolutionary algorithms, PSO does not use the "survival" concept. This is because all particles are kept "alive" throughout the algorithm execution time, and their survival is threatened.

In the PSO algorithm, the particles move around in the D-dimensional search space. The *i*th particle is represented as $X_i = \left(x_i^1, x_i^2, ..., x_i^D \right)$. The positions of individual particles are adjusted according to their previous best positions and the neighborhood best or the global best. The best previous position of the *i*th particle is recorded and represented as $P_i = \left(p_i^1, p_i^2, ..., p_i^D \right)$. The index of the best particle among all the particles in the population is represented by the symbol g. The rate of the position change (velocity) for particle *i* is represented as $V_i = \left(v_i^1, v_i^2, ..., v_i^D \right)$.

The searching procedure based on this concept can be described by

$$v_i^d(t+1) = \omega v_i^d(t) + c_1\phi\left(p_i^d(t) - x_i^d(t) \right) + c_2\phi'\left(p_g^d(t) - x_i^d(t) \right) \quad (5)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t) \quad (6)$$

where $\phi \; and \; \phi'$ are two different random numbers between 0 and 1; constants c_1 and c_2 are weight factors. Low values of c_1 and c_2 permit particles to travel far from the goal region before being togged back. On the other hand, high values result in rapid forward or backward movements, from the goal region. Eq. (6) is similar to a mutation operation, the PSO algorithm is similar to the evolutionary programming algorithm since neither algorithm performs a crossover operation.

From Eq. (5), the velocity of a particle is determined by three factors:

- $v_i^d(t)$, which serves as a momentum term to prevent excessive oscillations in search direction.
- $c_1\phi\left(p_i^d(t) - x_i^d(t) \right)$, referred to as the cognitive component. This component represents the distance that a particle is from the best solution, $p_i^d(t)$, found by itself. The cognitive component represents the natural tendency of individuals to return to environments where they experienced their best performance.
- $c_2\phi'\left(p_g^d(t) - x_i^d(t) \right)$, referred to as the social component. This component represents the distance from a (given) particle to the best position found by its neighborhood. It represents the tendency of individuals to follow the success of other individuals.

In [18] the author proposed the parameter ω into the PSO equation to improve its performance. The appropriate selection of inertia weight ω in Eq. (7) provides a balance between the global and local positions. As originally developed, ω often decreases linearly from about 0.9 to 0.4 during a run. Generally ω is defined by

$$\omega(t+1) = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{t_{\max}} t \quad (7)$$

SA algorithm is a popular local search meta-heuristic used to address discrete and continuous optimization problems. The interest began with the work of Kirkpatrick [23] and Cemy [9]. They showed how a model for simulating the annealing of solids, as proposed by Metropolis [28], could be used for problem optimization, where the objective function to be minimized corresponds to the energy of the state of the solid. In fact Cemy [9] was one of the first who applied SA over classical TSP. In [7] a Stochastic Annealing algorithm has been proposed in order to solve a PTSP. In there the authors proposed a proof-of-concept stochastic simulated annealing for the PTSP, in which the annealing schedule is controlled by the sampling error of the cost estimation. Unlike that approach, our work does not use the concept of error in the SA heuristics, it is used only for controlling the behavior of PSO

algorithm. Furthermore, in [7] the stochastic annealing approach was only tested on very small instances, and its performance has not been evidenced in medium and large size instances. Several other implementations of SA can be found in the literature. In [2], two estimation-based SA algorithms were developed using different acceptance criterion. In [16], the authors use the powerful global search capability of GA and the powerful partial search capability of SA, an improved SAGA is proposed in order to solve an specific problem in mining. Other SA hybridization can be found in [30, 31].

As mentioned above, SA is based on the Metropolis acceptance criterion [28], which models how a thermodynamic system moves from the current solution (state) $\omega \in \Omega$ to a candidate solution $\omega' \in N(\omega)$, in which the energy content is minimized. The candidate solution ω' is accepted as the current solution based on the acceptance probability [17]. Below, an abstract is presented about main characteristics of the SA algorithm, extracted from [17]. For more details, Eglese [12] and Ingber [18] give a complete overview.

$$P\begin{cases}\exp\left[\dfrac{-\left(f(\omega')-f(\omega)\right)}{t_k}\right] & if \quad f(\omega')-f(\omega)>0\\ 1 & if \quad f(\omega')-f(\omega)\le 0\end{cases} \tag{8}$$

Where P is the probability of accept ω'as a next solution. Define t_k as a temperature parameter in iteration k, such that

$$t_k > 0 \ \textit{for all k and} \ \lim_{k\to+\infty} t_k = 0 \tag{9}$$

This acceptance probability is the basic element of the search mechanism in simulated annealing. If the temperature is reduced sufficiently slowly, then the system can reach equilibrium (steady state) at each iteration k. Let $f(\omega)$and $f(\omega')$denote the energies (objective function values) associated with the solutions $\omega \in \Omega$ and $\omega' \in N(\omega)$ respectively. This equilibrium follows the Boltzmann distribution, which can be described as the probability of the system being in state $\omega \in \Omega$ with energy $f(\omega)$ at temperature T such as

$$P\left\{\frac{\exp\left[-f(\omega)/t_k\right]}{\sum_{\omega''\in\Omega}\exp\left[-f(\omega'')/t_k\right]}\right. \tag{10}$$

Hybrid PSO and SA Algorithm

As previously mentioned, PSO and SA algorithms have been applied to both TSP and PTSP problems separately. Recently, an hybrid algorithm of PSO-SA was used in order to solve a classical TSP [14], but there is no evidence in the literature about the application of an hybrid PSO-SA algorithm applied to PTSP. In [14], the fast optimal search ability of PSO and the probabilistic diversification property of SA are combined. The main difference between our algorithm and the proposed in [14] is the frequency in the use of SA algorithm for increasing the diversification level: In [14] SA only is used in the first particle search. By the other side, our PSO-SA algorithm uses the SA algorithm along all PSO algorithm iterations. Moreover, in our approach a simple sort algorithm is used, which performs a more efficient particle movement. Fan and Fang [15] developed another hybrid algorithm of PSO and SA named Niche Particle Swarm Optimization (NPSO) algorithm, which integrates SA and niche technique into PSO, using the rapid local search ability of PSO and global convergence of SA.

The SA algorithm is used in order to avoid being trapped into local minimum and to increase the diversity of particles.

The PSO-SA algorithm defines, as parameters, a NxN integer matrix (with N = #edges) which contains the distance between edge i and j. A PxN integer matrix called *Particle* was implemented. The P value corresponds to the number of particles that will be used in the algorithm. Each row of this matrix corresponds to the vision of the particle about each city and its neighbors. In fact, each row is a feasible solution for the PTSP. Each row can be considered as a cycle vector.

The step 1 of the hybrid algorithm is the initialization of the particles. For that, each particle (row in the *Particle* matrix) is initialized using a random value for each position. The unique constraint in this assignation is that the value is in the [0, N-1] interval and cannot be repeated in the same row.

Then, SA is introduced in order to obtain a good solution from each particle. For this, the energy function E was considered, such that $f(S_i)$ represents the length of the tour in particle S_i. Then, $\Delta f = f(S_{new}) - f(S_{old}) = \Delta E$ is defined, which corresponds to the energy gap between the particle (S_{old}) in the swarm and its neighbor (S_{new}). This neighbor is generated using a simple random swap move between two edges. Figure 2 shows this movement.

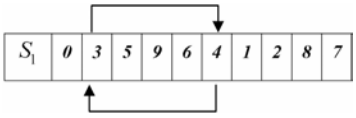

Figure 2. SA neighborhood movement.

The acceptance or rejection of the neighbor and update of particle tour depends on the following expression, which corresponds to our implementation of Eq. (11)

$$P \begin{cases} \exp\left[\dfrac{-\Delta f}{t_k}\right] & if \quad \Delta f > 0 \\ 1 & if \quad \Delta f \le 0 \end{cases} \tag{11}$$

Once each particle finds its best neighbour tour $S_i^{bestNeighbor}$ (using SA strategy) the best particle is selected from the Particle matrix, $S_{global}^{bestNeighbor}$. Then, each particle S_i, in the Particle matrix, is updated applying crossing operation on both $S_i^{bestNeighbor}$ and $S_{global}^{bestNeighbor}$. Finally, the temperature T is calculated using the following equation

$$T = \alpha^i T_0 + T_\lambda \tag{12}$$

where cooling coefficient α is a random constant in a range [0, 1], *i* is the number of iteration so far, T_0 is the initial temperature and T_λ is a limit value for end criterion.

As mentioned above, the main difference between our algorithm and the proposed by Fang in [14] is the frequency in the use of SA algorithm in order to increase the diversification level. In [14], SA is only used in the first particle search. By the other side, our PSO-SA algorithm uses the SA algorithm along all PSO algorithm iterations. Moreover, in our algorithm a simple sort strategy was used for the edges when the S_i particle is updated. In [14] the edges are considered in the same order that they have in the array (particle). In our case, the well known bubble sort algorithm was used for ordering the edges from the particle S_i: first the edges nearest the selected edge and finally the edges farthest the selected edge. The most important improvements related with this difference are the execution time and the quality of the solution obtained by the PSO-SA algorithm.

4. Experiments and Computational Results

This section shows the most relevant results obtained with our Hybrid PSO-SA algorithm for PTSP. First, a comparison among the hybrid algorithm with the PSO algorithm is provided. Then, the results obtained by the hybrid PSO-SA are shown for both well known PTSP instances obtained from TSPLIB[37]: eil101 and KroA200. These instances have 101 and 200 edges respectively and consider Euclidean distance between cities. The instances supply the coordinates of each city. These instances were solved using an homogeneous probability of {0.25; 0.50; 0.75} and compared with the results presented in [8].

Figure 3 shows the improvement obtained by the SA strategy on the PSO algorithm.

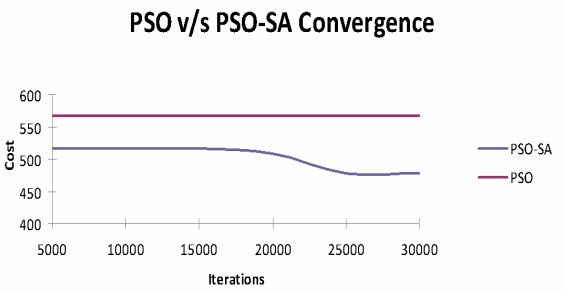

Figure 3. Comparison between PSO and PSO-SA algorithms for eil101 instance.

As can be seen, the PSO algorithm converges rapidly and is trapped on a local minimum being unable to escape from minimum. However, when SA strategy is included, the algorithm can exit from this local minimum and explore other surfaces, getting better solutions than those obtained by the simple PSO algorithm. Table I shows the behaviour of the algorithm when the number of particles vary. As Table I shows, for most of the instances the best result was obtained using 40 particles. Furthermore, the results show the stability of the PSO-SA algorithm which has a standard deviation of only 12.135 for probability of 0.5, and 17.2135 for probability of 0.75, which

corresponds to 1.94% and 2.36% respectively in relation with the mean (for eil101 instance).

guidance omits the information about the probability of the current or the target city

Table I. Behaviour of the PSO-SA Algorithm when the Number of Particles Vary

	50 Iterations			75 Iterations			100 Iterations		
Prob.	10 part	20 part	40 part	10 part	20 part	40 part	10 part	20 part	40 part
0.5	654.29	624.98	618.66	609.76	622.13	623.05	624.13	616.26	626.27
0.5	617.98	613.61	625.3	633.95	620.05	605.02	608.29	622.9	615.64
0.5	608.42	592.73	614.71	627.8	628.99	627.32	589.05	619.4	621.17
0.5	618.88	624.31	622.97	663.81	600.93	621.68	628.96	624.35	610.56
0.5	627.28	602.23	622.81	627.12	627.08	627.77	627.98	620.53	611.99
0.5	622.54	619.19	607.56	624.79	623.28	613.65	626.52	596.93	599.6
0.5	624.25	625.79	614.71	598.59	623.72	603.23	615.38	618.9	604.22
0.5	614.91	627.6	613.65	607.16	596.11	595.29	617.91	627.21	625.98
0.5	624.25	618.5	613.41	611.92	616.47	617.95	626.74	627.83	597.89
0.5	626.67	620.78	625.37	617.05	622.54	610.28	625.58	628.51	618.26
0.75	695.98	738.59	703.08	701.34	708.79	670.21	686.22	688.99	674.89
0.75	706.64	689.25	678.81	696.1	657.04	709.2	723.04	703.55	696.52
0.75	750.17	700.29	689.85	722	677.71	710.02	739.55	686.97	699.21
0.75	716.57	695.51	720.83	711.55	693.79	685.25	712.76	696.82	672.56
0.75	728.97	715.8	693.2	733.55	691.38	698.8	696.76	718.01	684.35
0.75	735.3	717.22	723.55	695.95	731.91	674.44	692.01	680.62	687.34
0.75	724.61	732.24	698.17	701.12	723.77	710.23	736.74	674.34	683.63
0.75	727.21	679.29	724.43	741.21	691.5	677.55	708.15	689.38	681.08
0.75	740.19	701.42	720.14	747.96	711.43	692.16	683.53	730.47	695.99
0.75	746.81	683.71	692.16	710.57	732.55	712.3	700.85	670.21	685.35

The main factor to explain this behaviour is that the SA algorithm allows the PSO algorithm to exit from local optimal solutions and to increase the exploration level of PSO algorithm. Furthermore, as depicted in Table I, the increase in the number of iterations causes a decrement in the standard deviation. This situation can be explained by the increase in the exploration level obtained by the greater number of iterations.

Table II shows the values raised for the PSO-SA algorithm for two instances of PTSP, compared with 4 different algorithms. The first three algorithms were developed in [8]. The first one, called the Depth-based heuristic, is based on increasing overall solution quality implied by a local decision during tour construction. The next one heuristic takes the angle between adjacent edges of the tour into consideration and is called Angle-based heuristic [8]. The last one is called TSP-ACO, which is an adaptation of ACO algorithm for TSP that uses only the distance as a heuristic

actually being a part of the later realization of the problem instance [8].Furthermore, as in [8], has been included in the comparison one of the currently best performing heuristics: HS/1-Shift. Basically, HS/1-Shift is deterministic and terminates when it can no longer find a 1-Shift improving the solution [8]. In addition, the computational time of HS/1-Shift is negligible.

Our algorithm behaved in a stable way at all instances; however it did not improve the best results reported in [8]. Despite the above, in several instances the hybrid algorithm improved the best solution raised by the HS-1Shift algorithm.

Table II. Comparison between PSO-SA algorithm with other heuristics for PTSP

Instance	Prob	PSO-SA	TSP-ACO	Angle-ACO	HS-1Shift	Depth-ACO	Dif. (%)
Eil101	0.25	328.41	325.903	325.071	321.732	322.230	2.1
	0.5	470.21	467.441	463.57	463.360	460.563	2.1
	0.75	570.93	567.076	564.710	572.110	564.036	1.2
KroA200	0.25	18094.18	17816.2	17719.7	18517.0	17574.6	3.0
	0.5	23877.46	23461.2	23341.5	23999.8	23327.8	2,4
	0.75	27163.87	27205.6	27179.9	30908.1	27126.1	1,0

These results demonstrate the potential of the hybrid PSO-SA algorithm in order to solve SCOPs, in general, and PTSP specifically. In this sense, it is worth noting that the best results for these instances have been obtained with the Ant Colony Optimization (ACO) based algorithm which has demonstrated its effectiveness in several deterministic routing problems. On the other hand, PSO based algorithms do not get their best performance for these kind of problems. Despite the above, the hybrid PSO-SA algorithm obtains promising results. The *Dif.* column shows how our PSO-SA algorithm is quite close to the best solution found for each instance. Additionally, this column shows the stability of the algorithm, which has not considerable variations across the different instances.

Table II shows how the PSO-SA algorithm reaches its best performance (compared with the best solution) when probability $p = 0.75$. This situation can be explained by the diversification strategy. This is because when probability p increases, the search space also increases.

5. Conclusions

The results showed an improvement over previously obtained solutions. That improvement is obtained by the SA algorithm which supplies an additional diversification strategy to the PSO based algorithm. This hybridization had been implemented earlier for several deterministic combinatorial optimization problems. However, no evidence in the literature was found about other PSO-SA hybrid algorithm applied over PTSP or any other SCOP. In addition, a simple sort strategy was implemented using the bubble sort algorithm, which obtained important improvements in both execution time and cost. This behaviour confirms that simple PSO algorithm needs of some diversification strategy in order to exit of local optimal solution. This paper shows that PSO with SA diversification strategy can solve a SCOP obtaining good solutions. Other diversification strategies, as to use other local search heuristic e.g. Tabu Search, could be implemented in order to improve current results. The preliminary results were improved implementing a neighbourhood insertion analysis strategy. Comparative experiments were made between the proposed algorithm and simple PSO. The computational results validate the effectiveness of the proposed approach for SCOPs.

As future work, we expect to apply our hybrid PSO-SA algorithm to other SCOPs, such as a probabilistic VRP and its variants, among others. On the other hand, improvements to the hybrid algorithm can be made; in this sense, other SA implementations can be used in order to supply better diversification strategies to the PSO based algorithm. In addition, a more complex sorting algorithm, e.g. quick sort, could be implemented for long size SCOPs, in order to minimize the execution time.

REFERENCES

1. ANGELINE, P. J., **Using Selection to Improve Particle Swarm Optimization**. IEEE Int. Conf. on Computational Intelligence (ICEC'98), 1998, pp. 84-89.
2. BALAPRAKASH, P., M. BIRATTARI, T. Stützle, M. Dorigo, **Estimation-based Metaheuristics for the Probabilistic Traveling Salesman Problem**. Computers & Operations Research, vol. 37, no. 11, 2010, pp. 1939-1951.
3. BALAPRAKASH, P., M. BIRATTARI, T. STÜTZLE, Z. YUAN, M. DORIGO, **Estimation-based Ant Colony Optimization and Local Search for the**

Probabilistic Traveling Salesman Problem. Swarm Intelligence, vol 3, 2009, pp. 223-242.

4. BERTSIMAS, D., P. JAILLET, A. ODONI, **A Priori Optimization**. Operations Research, vol. 38, no. 6, 1990, pp. 1019–33.
5. BIANCHI, L., L. M. GAMBARDELLA, M. DORIGO, **Solving the Homogeneous Probabilistic Traveling Salesman Problem by the ACO Metaheuristic**. Ant Algorithms (ANTS'02) LNCS vol. 2463, 2002, pp. 176-187.
6. BIANCHI, L., L. M. GAMBARDELLA, M. DORIGO, **An Ant Colony Optimization Approach to the Probabilistic Traveling Salesman Problem**. International Conference on Parallel Problem Solving from Nature (PPSN'02), 2002, pp. 883-892.
7. BOWLER, N. E., T. M. A. FINK, R.C. BALL, **Characterization of the Probabilistic Traveling Salesman Problem**. Physical Review part E, vol. 68, 2003, pp. 36703–36710.
8. BRANKE, J., M. GUNTSCH, **Solving the Probabilistic TSP with Ant Colony Optimization**. J. of Mathl. Modelling and Algorithms, vol. 3, 2004, pp. 403-425.
9. CEMY, V., **Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm**, Journal of Optimisation Theory and Applications, vol. 45, 1985, pp. 41-51.
10. EBERHART, R. C., R. W. DOBBINS, P. SIMPSON, **Computational Intelligence PC Tools**. Boston: Academic Press. ISBN: 0-12-228630-8. 1996.
11. EBERHART, R.C., J. KENNEDY, **A New Optimizer Using Particle Swarm Theory**. Proceedings Sixth International Symposium on Micro Machine and Human Science (MHS'95), 1995, pp. 39-43.
12. EGLESE, R. W., **Simulated Annealing: A Tool for Operational Research**. European Journal of Operational Research, vol. 46, 1990, pp. 271-281.
13. DURÁN O., N. RODRIGUEZ, L. CONSALTER, **Collaborative Particle Swarm Optimization with a Data Mining Technique for Manufacturing Cell Design**. Expert Systems with Applications, vol. 37, 2010, pp. 1563–1567
14. FANG, L., P. CHEN, S. LIU., **Particle Swarm Optimization with Simulated Annealing for TSP**. Proc. of the 6th Intl. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, (WSEAS '07), 2007, pp. 206-210.
15. FAN, X., X. FANG, **On Optimal Decision for QoS-Aware Composite Service Selection**. Information Technology Journal, vol. 9, no. 6, 2010, pp. 1207-1211
16. XIA, Y., G. ZHANG, S. NIE, Y. BU, Z. ZHANG, **Optimal Control of Cobalt Crust Seabed Mining Parameters Based on Simulated Annealing Genetic Algorithm**. Journal of Central South University of Technology, vol. 18, no. 3, pp. 650-657
17. HENDERSON, D., S. H. JACOBSON, A. W. JOHNSON, **The Theory and Practice of Simulated Annealing**. Handbook on Metaheuristics, F. Glover and G. Kochenberger, Editors, Kluwer Academic Publishers, Norwell MA: 287-320. ISBN: 1-4020-7263-5. 2003.
18. INGBER, L., **Simulated Annealing: Practice Versus Theory**. Mathl. Comput. Modelling, vol. 18, no. 11, 1993, pp. 29-57.
19. JAILLET, P., **A Priori Solution of a Traveling Salesman Problem in which a Random Subset of the Customers are Visited**. Operations Research, vol. 36, no. 6, 1988, pp. 929-936.
20. KENNEDY, J., R. C. EBERHART, **Particle Swarm Optimization**. Proc. of the IEEE Intl. Conf. on Neural Networks, vol. 4, 1995, pp. 1942-1948.
21. KENNEDY, J., R. C. EBERHART, Y. SHI, **Swarm Intelligence**. The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann, San Francisco-California. 2001.
22. KENNEDY, J., **The particle swarm: social adaptation of knowledge**. Proc. IEEE Intl. Conf. on Evolutionary Computation (CEC'97), 1997, pp. 303-308.
23. KIRKPATRICK, S., C. D. GELATT, M. P. VECCHI, **Optimization by simulated annealing**, Science, vol. 220, 1983, pp. 671-680.

24. KULIC, F., D. MATIC, B. DUMNIC, V. VASIC, **Optimal Fuzzy Controller Tuned by TV-PSO for Induction Motor Speed Control**. Advances in Electrical and Computer Engineering, vol. 11, no.1, 2011, pp. 49-54.

25. LIU, Y., **Different Initial Solution Generators in Genetic Algorithms for Solving the Probabilistic Traveling Salesman Problem**. Applied Mathematics and Computation, vol. 216, no. 1, 2010, pp. 125-137.

26. MARINAKIS, Y., A. MIGDALAS, P. M. PARDALOS, **Expanding Neighborhood search-GRASP for the Probabilistic Traveling Salesman Problem**. Optimization Letters, vol. 2, no. 3, 2008, pp. 351-361.

27. MARINAKIS, Y., M. MARINAKI, **A Hybrid Multi-Swarm Particle Swarm Optimization algorithm for the Probabilistic Traveling Salesman Problem**. Computers & Operations Research, vol. 37, no. 3, 2010, pp. 432-442.

28. METROPOLIS, N., A. ROSENBLUTH, M. ROSENBLUTH, A. TELLER, E. TELLER, **Equation of State Calculations by Fast Computing Machines**. Journal of Chemical Physics, vol. 21, 1953, pp. 1087-1092.

29. NIKNAM, T., **An Approach Based on Particle Swarm Optimization for Optimal Operation of Distribution Network Considering Distributed Generators**. IEEE 32nd Annual Conf. on Industrial Electronics (IECON' 06), 2006, pp. 633-637.

30. NIKNAM, T., J. OLAMAEI, B. AMIRI, **A Hybrid Evolutionary Algorithm Based on ACO and SA for Cluster Analysis**. Journal of Applied Sciences, vol. 8, 2008, pp. 2695-2702.

31. RAHMANI, K., M. MOLAVI, B. NADERI, M. SOLTANI, **A Hybridization of Simulated Annealing and Electromagnetic Algorithms for Flowshop Problems with Skipping Probability**. Journal of Applied Sciences, vol. 9, 2009, pp. 2438-2444.

32. SAADI S., M. BETTAYEB, A. GUESSOUM, **Optimal Approach for Neutron Images Restoration using Particle Swarm Optimization Algorithm with Regularization**. Journal of Applied Science, vol. 10, no. 7, 2010, pp. 517-525.

33. SHI, X. H., Y. C. LIANG, H. P. LEE, C. LU, Q. X. WANG, **Particle Swarm Optimization-based Algorithms for TSP and Generalized TSP**. Information Processing Letters, vol. 103, 2007, pp. 169–176.

34. SHI, Y., R. C. EBERHART, **Empirical Study of Particle Swarm Optimization**. Proc. of the IEEE Congress on Evolutionary Computation (CEC'99), 1999, pp. 1945–1950.

35. WANG, Q., L. XIONG, H. LIU, J. LIANG, **Improved PSO for TSP based on the Information Communication and Dynamic Work Allocation**. Asian Journal of Information Technology, vol. 5, no.11, 2006, pp. 1191-119.

36. BOUKEF, H., M. BENREJEB, P. BORNE, **Flexible Job-shop Scheduling Problems Resolution Inspired from Particle Swarm Optimization**. Studies in Informatics and Control, vol 17, no. 3, 2008, pp. 241-252.

37. TSPLIB:http://www2.iwr.uni-heidelberg.de /groups/comopt/software/TSPLIB95/index. html, 2011