Scatter Search Particle Filter to Solve the Dynamic Travelling Salesman Problem

Juan José Pantrigo[1], Abraham Duarte[1], Ángel Sánchez[1], and Raúl Cabido[1]

[1]Universidad Rey Juan Carlos, c/ Tulipán s/n
28933 Móstoles, Spain
{j.j.pantrigo, a.duarte, an.sanchez}@escet.urjc.es

Abstract. This paper presents the Scatter Search Particle Filter (SSPF) algorithm and its application to the Dynamic Travelling Salesman Problem (DTSP). SSPF combines sequential estimation and combinatorial optimization methods to improve the execution time in dynamic optimization problems. It allows obtaining new high quality solutions in subsequent iterations using solutions found in previous time steps. The hybrid SSPF approach increases the performance of general Scatter Search (SS) metaheuristic in dynamic optimization problems. We have applied the SSPF algorithm to different DTSP instances. Experimental results have shown that SSPF performance is significantly better than classical DTSP approaches, where new solutions of derived problems are obtained without taking advantage of previous solutions corresponding to similar problems. Our proposal reduces execution time appreciably without affecting the quality of the estimated solution.

1 Introduction

Dynamic optimization problems are characterized by an initial problem definition and a collection of "events" over the time. An event defines some changes on the data of the problem [1]. Therefore the optimization method needs from adaptive strategies for these changing conditions. In these problems, a key question is how to use information found in previous time steps to obtain high quality solutions in subsequent ones, without restarting the computation from scratch.

Dynamic optimization problems play an important role in industrial applications, such as transportation, telecommunications and manufacturing [1]. Surprisingly, compared to the amount of research undertaken on static optimization problems, relatively little work has been devoted to dynamic problems [2][3].

Unlike static problems, dynamic ones often lack well defined optimization functions, standard benchmarks or criteria for comparing solutions [1][4][2][3]. Nowadays, main used strategies have been specific heuristics [1] and manual procedures [5][2]. Traditionally, metaheuristics using constructive methods such as Ant Systems [6][7] and population-based metaheuristics such as evolutionary algorithms [8] have been applied to dynamic problems.

The filtering problem deals with updating the present state of knowledge and predicting with drawing inferences about the future state of the system [9]. Sequential Monte Carlo algorithms (also called particle filters) are a special class of filters in which theoretical distributions on the state space are approximated by simulated random measures (also called particles) [9].

We propose a new approach, called Scatter Search Particle Filter (SSPF) to solve the Dynamic Travelling Salesman Problem. SSPF combines sequential estimation (Particle Filter) [9][10] and metaheuristic methods (Scatter Search) [11] in two different stages. In the Particle Filter (PF) stage, a particle set is propagated and updated to obtain a new particle set. In the Scatter Search (SS) stage, some solutions from the particle set are selected and combined to obtain new optimized solutions.

This paper considers a dynamic variant of the Travelling Salesman Problem [12] in which "distances" among cities vary over the time. In dynamic problems, meta-heuristics based approaches tend to restart the search method after events or to use the best solutions found in previous time steps. In the first approach, the derived problem is processed as unrelated with respect to its origin problem. It implies that useful information can be rejected, increasing the computation time. As a consequence, a reasonable trade-off between the analysis of the prior problem solution and actual problem computational effort must be found. Experimental results have shown that the proposed algorithm has successfully been applied to different DTSP instances.

2   The Dynamic Travelling Salesman Problem

The Travelling Salesman Problem (TSP) consists of finding the shortest tour connecting a fixed number of locations (cities), visiting each city exactly once [12]. This problem can be represented by a graph $G = \{V, E, W\}$, where V is a set of vertex representing the cities, E is a set of edges which model the paths connecting cities and W is a symmetric matrix of weights. We suppose that there is an edge jointing every pairs of cities. Weighs $w_{ij} \in W$, attached to edges $(i,j) \in E$ represent the distance between cities $i, j \in V$. The TSP can be described as the problem of finding a Hamiltonian circuit with minimum length in the graph G [8]. The TSP belongs to the NP-hard class [13]. It is one of the most considered problems in Combinatorial Optimization and several approaches have been used to solve it. In the public library TSPLIB [14] sample instances for the static TSP can be found.

The Dynamic Travelling Salesman Problem (DTSP) is a generalization on TSP where G is time-dependent. This problem has got several practical applications such as traffic jams [12] or fluctuating set of active machines [15]. Two different varieties of Dynamic Travelling Salesman Problem exist in the literature. The first one consists of inserting or deleting cities into a given problem instance [16][17]. A different approach [12] consists of keeping constant the number of cities, allowing distance changing among them. It can be applied to describe traffic jams and motorways. In this context, good solutions may not be optimal after changes and the salesman needs to be re-routed. In this paper, we focus on the second approach.

Ant Systems (AS) and Evolutionary Computation (EC) have been the most common applied metaheuristics to solve the DTSP. These techniques should be able

to adjust their solutions under changing environments. In [12][15][17] different AS implementations applied to DTSP can be found. The main reason for using AS to dynamic problems is based on the pheromone concept. Pheromone can be exploited as reinforcement, and therefore, as a way to transferring knowledge. When a change is detected a partial decomposition-reconstruction procedure is performed over old solutions [1]. This procedure determines which elements of ant's solutions must be discarded to satisfy the feasibility of the new conditions.
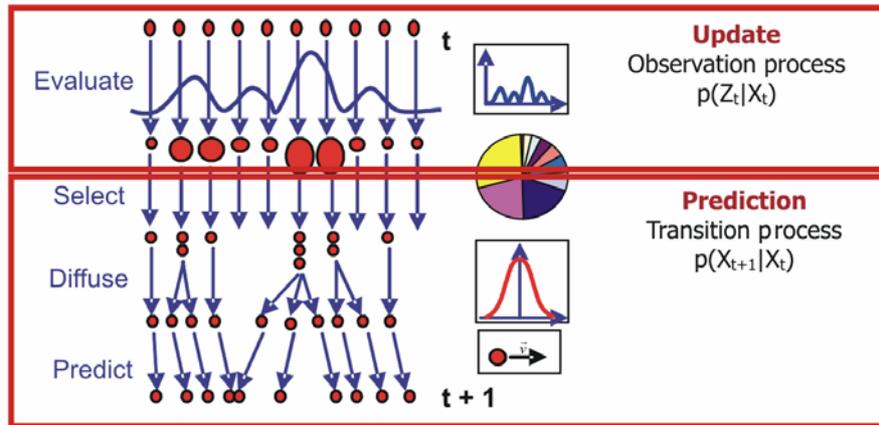

Fig. 1. Particle Filter scheme

3   Particle Filter Framework

Sequential Monte Carlo algorithms (also called Particle Filters) are filters in which theoretical distributions on the state space are approximated by simulated random measures (called particles) [9]. The state-space model consists of two processes: (i) an observation process $p(Z_t|X_t)$, where X denotes the system state vector and Z is the observation vector, and (ii) a transition process $p(X_t|X_{t-1})$. Assuming that observations $\{Z_0, Z_1, \ldots, Z_t\}$ are known, the goal is to recursively estimate the posterior pdf $p(X_t|Z_t)$ and the new system state $\{\chi_0, \chi_1, \ldots, \chi_t\}$ at each time step. In Sequential Bayesian Modelling framework, posterior pdf is estimated in two stages:

(i) Evaluation: posterior pdf $p(X_t|Z_t)$ is computed at each time step by applying Bayes theorem, using the observation vector Z_t:

$$p(X_t \mid Z_t) = \frac{p(Z_t \mid X_t)p(X_t \mid Z_{t-1})}{p(Z_t)} \tag{1}$$

(ii) Prediction: the posterior pdf $p(X_t|Z_{t-1})$ is propagated at time step t using the Chapman-Kolmogorov equation:

$$p(X_t \mid Z_{t-1}) = \int p(X_t \mid X_{t-1})p(X_{t-1} \mid Z_{t-1})dX_{t-1} \tag{2}$$

A predefined system model is used to obtain an updated particle set.

In Figure 1 an outline of the Particle Filter scheme is shown. The aim of the PF algorithm is the recursive estimation of the posterior pdf $p(X_t|Z_t)$, that constitutes the complete solution to the sequential estimation problem. This pdf is represented by a set of weighted particles $\{(\boldsymbol{x}_t^0, \pi_t^0)\dots(\boldsymbol{x}_t^N, \pi_t^N)\}$.

PF starts by setting up an initial population X_0 of N particles using a known pdf. The measurement vector $\boldsymbol{Z}_t$ at time step t, is obtained from the system. Particle weights $\boldsymbol{\Pi_t}$ are computed using a weighting function. Weights are normalized and a new particle set X^*_t is selected. As particles with larger weight values can be chosen several times, a diffusion stage is applied to avoid the loss of diversity in X^*_t. Finally, particle set at time step $t+1$, X_{t+1}, is predicted using the motion model. A pseudocode of a general PF is detailed in [10][18]. Therefore, Particle Filters can be seen as algorithms handling the particles time evolution. Particles in Particle Filters move according to the state model and are multiplied or died according to their weights or fitness values as determined by the likelihood function [9].

4   Scatter Search

Scatter Search (SS) [19][20] is a population-based metaheuristic that provides unifying principles for recombining solutions based on generalized path construction in Euclidean spaces. In other words, SS systematically generates disperse set of points (solutions) from a chosen set of reference points throughout weighted combinations. This concept is introduced as the main mechanism to generate new trial points on lines jointing reference points. SS metaheuristic has been successfully applied to several hard combinatorial problems. A recent method review can be found in [20].
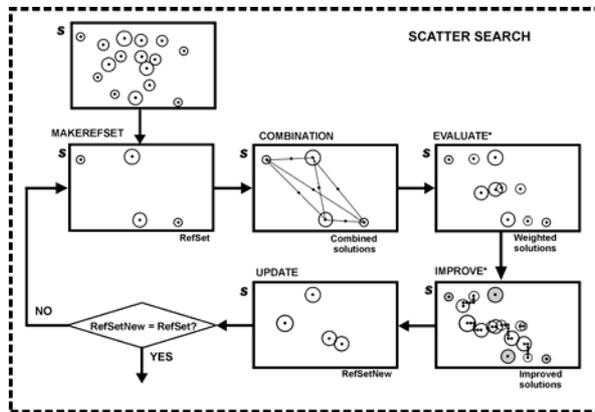

Fig. 2. Scatter Search scheme

In Figure 2 an outline of the SS is shown. SS procedure starts by choosing a subset of solutions (called *RefSet*) from a set *S* of initial feasible ones. The solutions in *RefSet* are the *h* best solutions and the *r* most diverse ones of *S*. Then, new solutions are generated by making combinations of subsets (pairs typically) from *RefSet*. The resulting solutions, called trial solutions, can be infeasible. In that case, repairing

methods are used to transform these solutions into feasible ones. In order to improve the solution fitness, a local search from trial solutions is performed. SS ends when the new generated solutions do not improve the quality of the *RefSet*.

5 Scatter Search Particle Filter

In our opinion, dynamic optimization problems deal with optimization techniques, but also with prediction tasks. This is due to the fact that the optimization method for changing conditions needs from adaptive strategies. Therefore, one key aspect is how to efficiently use important information found in previous time steps in order to find high quality solutions for new derived problems instances.

Usually in metaheuristics, two approaches can be used depending on the problem change rate. If it is high, each problem is tackled as a different one, so the computation is restarted from scratch. If change rate is low, the last solution or a set of the best solutions found are used as starting point in the new search. For instance, Genetic Algorithms use the previous population as initial set in the next time step. On the other hand, Ant Colony Optimization uses the previous pheromone deposition in each node as initial pheromone distribution of subsequent steps. The same idea can be extended to other metaheuristics. In Scatter Search, the *RefSet* obtained in the previous time step can be used as a new *RefSet* for the next one or *RefSet* could be improved with diverse solutions. Nevertheless, it is very important to make a decision of which information is propagated to the next time step. This is because it is possible that the search algorithm get stuck near local optimum. As a consequence, a reasonable trade-off between both restart from scratch and restart from previous optimum must be found. Therefore, it could not be appropriate to use optimization procedures in the prediction stage.

Analogously, sequential estimation algorithms like particle filters are well-suited in prediction stages, but they are not good enough for solving dynamic optimization problems. Optimization strategies performed with this kind of algorithms are usually very computationally inefficient.

Then, dynamic optimization problems need from both optimization and prediction tasks. The key question is how to hybridize these two kinds of algorithms to obtain a new one which combines both techniques. In this way, a novel hybrid algorithm called Scatter Search Particle Filter (SSPF) is presented to solve the Dynamic TSP.

5.1 Scatter Search and Particle Filter Hybridization

The Scatter Search Particle Filter (SSPF) algorithm is introduced in this paper to be applied to dynamic optimization problems. SSPF integrates both Scatter Search (SS) and Particle Filter (PF) frameworks in two different stages:

- In the Particle Filter stage, a particle set is propagated and updated to obtain a new one. This stage is focused on the evolution in time of the best solutions found in previous time steps. The aim for using PF is to avoid the loss of needed diversity in the solution set.
- In the Scatter Search stage, a fixed number of solutions from the particle set are selected and combined to obtain better ones. This stage is devoted to improve the quality of a reference subset of good solutions in such a way that the final solution is also improved.
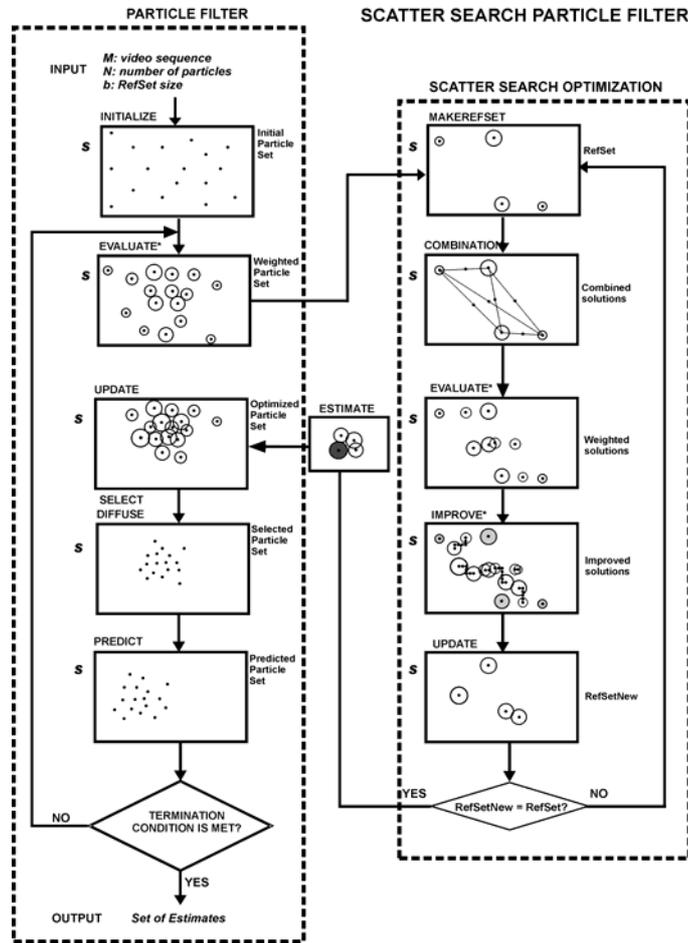
Fig. 3. Scatter Search Particle Filter scheme. Weight computation is required during EVALUATE and IMPROVE stages (*)

Figure 3 shows a graphical template of the SSPF algorithm. Dashed lines separate the two main components in the SSPF scheme: PF and SS optimization, respectively. SPF starts with an initial population of *N* particles drawn from a known pdf (Figure 3: INITIALIZE). Each particle represents a possible solution of the problem. Particle weights are computed using a weighting function (Figure 3: EVALUATE). SS stage is later applied to improve the best obtained solutions of the particle filter stage. A Reference Set (*RefSet*) is created selecting a subset of *b* (*b<<N*) particles from the particle set (Figure 3: MAKEREFSET). This subset is composed by the *b*/2 best solutions and the *b*/2 most diverse ones of the particle set. New solutions are generated and evaluated, by combining all possible pairs of particles in the *RefSet* (Figure 3: COMBINE and EVALUATE). In order to improve the solution fitness, a local search from each new solution is performed (Figure 3: IMPROVE). Worst

solutions in the *RefSet* are replaced when there are better ones (Figure 3: UPDATEREFSET). SS stage ends when new generated solutions *RefSetNew* do not improve the quality of the *RefSet*. Once the SS stage is finished, the "worst" particles in the particle set are replaced with the *RefSetNew* solutions (Figure 3: INCLUDE). Then, a new population of particles is created by selecting the individuals from particle set with probabilities according to their weights (Figure 3: SELECT and DIFFUSE). Finally, particles are projected into the next time step by following the update rule (Figure 3: PREDICT).

5.2 Scatter Search Particle Filter Main Features

The SSPF leads the search process to a region of the search space in which it is highly probable to find new better solutions than the initial computed ones. PF increases the performance of general SS in dynamic optimization problems by improving the quality of the diverse initial solution set S. In order to obtain the solution set $S(t+1)$ PF performs two tasks over the set $S(t)$: (i) selecting the best solutions and (ii) predicting new solutions from the best ones. Firstly, the selection procedure selects particles with larger weight values more likely than those with lower weights. Secondly, PF performs a prediction procedure over these best solutions to obtain the set $S(t+1)$. In this way, PF tackles with problem changes in time by predicting the best solution time evolution. As results, solutions in $S(t+1)$ will be closer to global optimum than another ones obtained randomly. On the other hand, a diffusion procedure is applied to the selected solutions to include diversity in the set $S(t+1)$.

Therefore, SSPF adapts computational load to problem constraints, by reducing the number of required evaluations of the particle weighting function. In this way, solutions in *RefSet* will be selected from a better solutions set. This is the main reason why SSPF reduces the required number of evaluations for the fitness function, and hence the computational load.

SS and PF are related in such a way that when the SS improves its performance, the PF performance also improves and vice versa. PF allows parameter tuning in order to adjust the quality and the diversity of the set S, used by SS. On the other hand SS improves the quality of the particle set allowing the better estimation of the pdf, by including *RefSet* solutions in the set S. This fact yields to a highly configurable algorithm. The main considered SSPF algorithm parameters are:

- The *size of the particle set N* is the number of particles in the particle set. These should be enough particles to support a set of diverse solutions, avoiding the loss of diversity in the particle set. Therefore, N influences on the performance of the SS stage. The value of N depends on the problem instance complexity.
- The *size of the reference set b* is the number of solutions in the *RefSet*. A typical b used in the literature is $b = 10$ [20][22].
- The diffusion stage is applied to avoid the loss of diversity in S. It is performed by applying a random displacement with *maximum amplitude A*. This amplitude is a measure of the diversity produced in the new particle set. Therefore, A influences the performance of the SS by tuning the diversity of the initial solution set, and hence, the diversity of the *RefSet*.

The SSPF is presented to be applied to hard dynamic optimization problems. In this kind of problems, it is usual to perform some preliminary experimentation in order to achieve the appropriate parameter values.

6 SSPF Implementation to DTSP Solving

In the Scatter Search Particle Filter (SSPF) implementation, solutions (particles) are represented as paths over cities. The number of particles N in the particle set S is chosen according to the problem size. Concretely, N varies from 100 in the 25-cities problem to 1000 in the 100-cities problem. The *RefSet* is created by selecting the 5 best solutions and the 5 most diverse ones in S.

DTSP is considered as an R-permutation problem [20], because relative positioning of the elements is more important than absolute positioning. Therefore, to find the most diverse solutions, the distance metric for *R-permutation problems* was used [20]. The distance between two solutions p and q for R-permutation problems is defined as:

$$d(p,q) = \text{number of times } p_{i+1} \text{ doesn't immediately follow } p_i \text{ in } q \text{ for } i=1,\ldots, n\text{-}1 \quad (3)$$

Voting method [20] has been used as combination procedure over all pairs of solutions in the *RefSet*. In this procedure, each reference solution votes for its first sector not included in the combined solution. The voting determines the element to be assigned to the next free position in the combined solution.

The *2-opt method* [21] was employed as improvement stage in the SS scheme (Figure 3). Given a solution, consider all pairs of edges connecting four different cities are considered. Removing two edges from the solution tour, there is a unique way of reconnecting the two remaining paths such that a new tour is obtained. If the new tour is shorter, then it replaces the old tour and the procedure is repeated until no such improvement is produced.

7 Experimental Results

To analyze the performance of the proposed algorithm, implementations of three evolutionary methods have been developed: Scatter Search, Evolutionary Algorithm and Scatter Search Particle Filter. The experiments were evaluated in an Intel Pentium 4 at 1.7 GHz and 256 MB RAM. All algorithms were coded in MATLAB 6.1, without optimization and by the same programmer. Different methods were applied to several instances of DTSP and results were compared. The following sections are devoted to describe used data, algorithms and obtained results.

7.1 Problem Instances

Unfortunately, as far as the authors know, there are no benchmarks for the DTSP. Thus, we generated synthetic and standard-based data. Synthetic data are composed by four different graph sequences, using 25, 50, 75 and 100 cities. Each sequence is composed by 10 different graphs. To know the value of the optimum, cities are
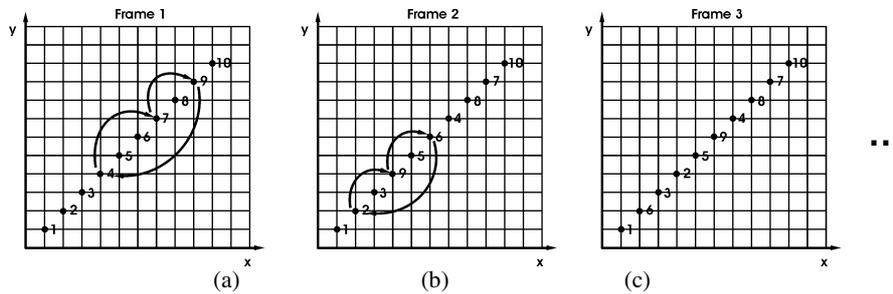
Fig. 4. Graph sequence generation process

located in the Euclidean plane along the diagonal as shown in Figure 4. In the first frame, cities are located in lexicographic order. Subsequent frames are generated by performing exchanges of cities in groups of three (see Figure 4). The average probability of node exchange in the considered graph sequences was $p_{change} = 0.15$.

Standard-based data are built as dynamic version of benchmarks from the public-domain library TSPLIB [14]. In particular, they are dynamic modifications of BAYG29, BERLIN51 and ST70 instances. We built each sequence starting from the original graph. Subsequent 4 graphs are obtained from the previous one introducing a perturbation in the actual location of each city according to a Gaussian distribution.

7.2   Algorithms Description

Different versions of Scatter Search and Evolutionary Algorithms were implemented. All of them use as stopping criteria "10^6 evaluations of fitness function or none improvement in the population". Solutions are coded as tours connecting all cities.

The two Scatter Search implementations were called SS1 and SS2. SS1 considers that each graph in the sequence is totally decoupled from the previous ones. Therefore, computation is restarted from scratch after graph changes. Basically, SS1 is based on Campos implementation described in [20][22].

In the second implementation (SS2) the graphs are supposed quite related. Thus, the *RefSet* obtained in the actual time step is used as a new *RefSet* for the next one. SS parameters *PopSize* and *b* for both implementations SS1 and SS2 were set to 100 and 10 respectively, as recommended in [22]. To obtain comparable results, we use the same *RefSet* composition, combination and improvement methods as in SSPF implementation (5-best and 5-diverse solutions in *RefSet*, voting method and 2-opt).

The implementation of the Evolutionary Algorithm (EA) performs the main stages of a standard genetic algorithm, including an improvement stage. The algorithm use voting method and 2-opt. EA parameters were set to *PopSize* = 100, crossover probability $p_c = 0.25$ and mutation probability $p_m = 0.01$ as recommended in [23]. Finally, improvement probability was set to $p_c = 0.25$.

7.3   Computational Testing

Experimental results are organized in three sections. In the first one the approach goodness in synthetic data is justified. Next section is devoted to the comparison between the SSPF and the two SS and the EA implementations.
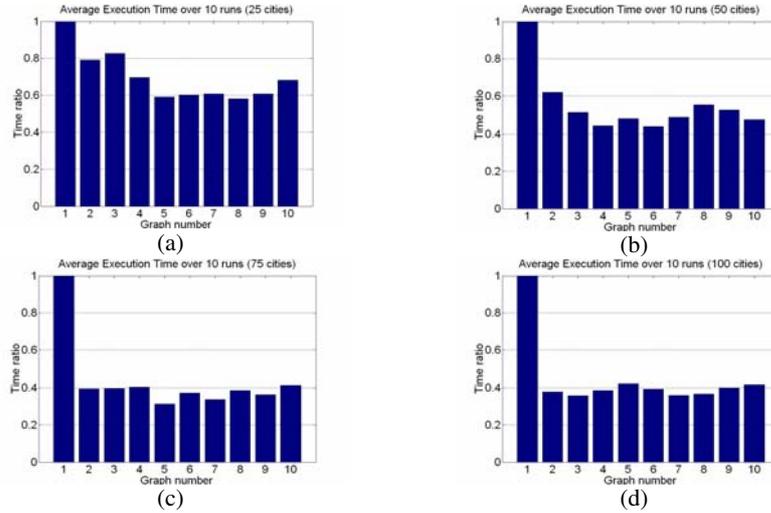
Fig. 5. Fitness function evaluations per graph in (a) 25, (b) 50, (c) 75 and (d) 100-cities problems

7.3.1 SSPF in Synthetic Data

Experimental results obtained by applying SSPF to synthetic data are presented in this section. In table 1, mean value of the execution time for the first graph is compared to the mean value of the execution time for the rest of the graph sequence.

The proposed strategy, based on Particle Filter and Scatter Search hybridization seems to be more advantageous than the classical SS one, in which an execution from scratch is performed. In this table, the column *Ratio* represents the average time SSPF improvement with respect to the corresponding time of the SS1 solution. As it can be seen, ratio between execution times is always in favor of SSPF algorithm.

Figure 5 shows the average fitness function evaluation per frame, over 10 runs of the same graph sequence. Each one is composed by 10 similar graphs. In this figure, relative execution time is represented for each frame. Elapsed time for the 2nd to 10th graphs (SSPF improvement) significantly reduces elapsed time for the first graph (SS approach) in all instances. Results show that SSPF achieves the best solution in all instances. Moreover, it is faster than SS1 implementation without loss of quality.

Table 1. Average fitness function evaluations values over 10 runs for each graph sequence

Nº of cities	Size of RefSet	Average Time SS	Average Time SSPF	Ratio
25	100	0.3×10^6	0.2×10^6	0.69
50	100	2.1×10^6	1.1×10^6	0.55
75	500	6.8×10^6	3.0×10^6	0.44
100	1000	14.7×10^6	6.6×10^6	0.44

7.3.2 SSPF Vs SS and EA in Standard-Based Data

This section presents a comparison between SSPF and two different implementations of SS. Results obtained by these algorithms (SS1, SS2, EA and SSPF) over all standard-based data (BAYG29, BERLIN52 and ST70) are presented in figure 6. Because of initial conditions and initial procedures performed are the same in SS1, SS2 and SSPF, solutions found in the first graph is exactly the same one. As the EA approach is different to the other ones, the solution and the time required to found this solution in the first graph are also unlike.
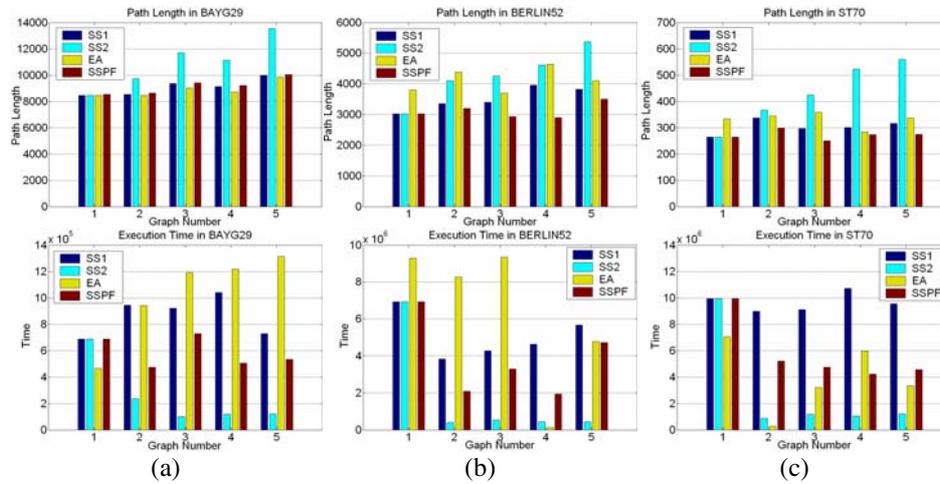

Fig. 6. Path length (upper row) and fitness function evaluations (lower row) using SS1, SS2, EA1 and SSPF in (a) BAYG29, (b) BERLIN52 and (c) ST70

Quality of the estimation performed by SS1 and SSPF are similar in subsequent graphs. However, execution time is significantly lower in SSPF approach, as explained in previous sections. In the SS2 implementation, the search procedure is trapped in a local optimum (maybe in the neighbourhood of the previous optimum). This yields SS2 achieves the lowest execution time, but with very poor quality. Finally, EA finds good quality solutions, but the time required to obtain them is larger than using SSPF. Table 2 resumes the main results obtained using different approaches. Execution time and path length demonstrate the performance of SSPF.

Table 2. Average execution time and path lengths of SS1, SS2, EA and SSPF over all instances

	SS1		SS2		EA		SSPF	
Cities	*Length*	*Time*	*Length*	*Time*	*Length*	*Time*	*Length*	*Time*
BAYG29	0.86×10^6	0.91×10^4	0.25×10^6	1.09×10^4	1.03×10^6	0.89×10^4	0.58×10^6	1.09×10^4
BERLIN52	5.07×10^6	3.51×10^3	1.75×10^6	$4..27 \times 10^3$	6.37×10^6	4.12×10^3	3.79×10^6	3.11×10^6
ST70	9.65×10^6	302.97	2.84×10^6	427.58	3.97×10^6	331.15	5.72×10^6	272.15

8   Conclusions

The main contribution of this work is the development of the Scatter Search Particle Filter (SSPF) algorithm. SSPF hybridizes the Scatter Search metaheuristic and the Particle Filter framework to solve dynamic problems. We have successfully applied the proposed SSPF algorithm to the Dynamic Travelling Salesman Problem (DTSP). Experimental results have shown that SSPF appreciably increases the performance of derived Scatter Search and Evolutionary Algorithm methods in a challenging dynamic optimization problem (DTSP), without losing quality in the estimation procedure. This improvement is more significant as the size of the problem increases.

References

1. Randall, M.: Constructive Meta-heuristics for Dynamic Optimization Problems. Technical Report. School of Information Technology. Bond University (2002)
2. Sadeh, N., Kott, A.: Models and Techniques for Dynamic Demand-Responsive Transportation Planning. Tech. Rept. TR-96-09, Carnegie Mellon University (1996)
3. Dror, M., and Powell, W.: Stochastic and Dynamic Models in Transportation. Operations Research, 41 (1993) 11-14
4. Beasley, J., Krishnamoorthy, M., Sharaiha, Y. and Abramson, D.: The displacement Problem and Dynamically Scheduling Aircraft Landings. Working paper, Available online at http://graph.ms.ic.ac.uk/jeb/displace.pdf (2002)
5. Beasley, J., Sonander, J. and Havelock, P.: Scheduling Aircraft Landings at London Heathrow using a Population Heuristic, Journal of the Operational Research Society, 52 (2001) 483-493
6. Glover, F., Kochenberger, G. A.: Handbook of metaheuristics. Kluwer (2002)
7. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. on Evolutionary Computation, 1(1) (1997) 53–66
8. Zhang-Can H.; Xiao-Lin H.; Si-Duo C.: Dynamic traveling salesman problem based on evolutionary computation. Proc. of Evolutionary Computation Conf, 2 (2001) 1283 - 1288
9. Carpenter, J., Clifford, P., Fearnhead, P.: Building robust simulation based filters for evolving data sets. Tech. Rep., Dept. Statist., Univ. Oxford, Oxford, U.K. (1999)
10. Arulampalam, M., et al.: A Tutorial on Particle Filter for Online Nonlinear/Non-Gaussian Bayesian Tracking. IEEE Trans. On Signal Processing, 50 (2) (2002) 174–188
11. Blum, C., Roli, A.: Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys, 35 (3) (2003) 268 - 308
12. Eyckelhof, C.J., Snoek, M.: Ant Systems for A Dynamic DSP: Ants Caught in a Traffic Jam. Proc. of ANTS02 Conference (2002)
13. Karp, R.M.: Reducibility among Combinatorial Problems. R. Miller and J. Thatcher (eds.): Complexity of Computer Computations. Plenum Press (1972) 85-103
14. Reinelt, G.: TSPLIB. University of Heidelberg. Available online at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/ (1996)
15. Guntsh, M., Middendorf, M.: Applying Population based ACO to Dynamic Optimization Problems. In Ant Algorithms, Proceedings of Third International Workshop ANTS 2002, LNCS 2463 (2002) 111-122

16. Guntsh, M., Middendorf, M., Schmeck, H.: An Ant Colony Optimization Approach to Dynamic TSP. In Proc. GECCO-2001 Conference, San Francisco, CA: Morgan Kaufmann Publishers (2000) 860-867
17. Guntsh, M., Middendorf, M.: Pheromone Modification Strategies for Ant Algorithms applied to Dynamic TSP. Lecture Notes in Computer Science, 2037 (2001) 213-222
18. Pantrigo, J.J., Sánchez, A., Gianikellis, K., Duarte, A.: Path Relinking Particle Filter for Human Body Pose Estimation. Lecture Notes in Computer Science 3138 (2004) 653-661
19. Glover, F.: A Template for Scatter Search and Path Relinking. LNCS 1363 (1997) 1-53
20. Laguna, M., Marti, R.: Scatter Search methodology and implementations in C. Kluwer Academic Publisher (2003)
21. Vizeacoumar, F.: TSP Implementation. Project report Combinatorial Optimization CMPUT – 670
22. Campos, V., Laguna, M. , Martí, R.: Scatter Search for the Linear Ordering Problem. *New Ideas in Optimization*. McGraw-Hill (1999)
23. Michalewitz, Z.: Genetic Algorithms + Data Structures = Evolution Programs Springer-Verlag, 1996.