Chapter 13
Low-Level Hybridization of Scatter Search and Particle Filter for Dynamic TSP Solving

Juan José Pantrigo and Abraham Duarte

Abstract. This work presents the application of the Scatter Search Particle Filter (SSPF) algorithm to solve the Dynamic Travelling Salesman Problem (DTSP). SSPF combines sequential estimation and combinatorial optimization methods to efficiently address dynamic optimization problems. SSPF obtains high quality solutions at each time step by taking advantage of the best solutions obtained in the previous ones. To demonstrate the performance of the proposed algorithm, we conduct experiments using two different benchmarks. The first one was generated for us and contains instances sized 25, 50, 75 and 100-cities and the second one are dynamic versions of TSPLIB benchmarks. Experimental results have shown that the performance of SSPF for the DTSP is significantly better than other population based metaheuristics, such as Evolutionary Algorithms or Scatter Search. Our proposal appreciably reduces the execution time without affecting the quality of the obtained results.

13.1 Introduction

Dynamic optimization problems are characterized by an initial problem definition and a collection of ”events” over the time. These ”events” define changes on the data of the problem [26]. Therefore, dynamic optimization methods arise from strategies to adapt for non-stationary conditions. In dynamic optimization 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. Many real-life problems belong to this category, particularly in transportation, telecommunications and manufacturing areas [26]. Surprisingly, compared to

Juan José Pantrigo · Abraham Duarte
Universidad Rey Juan Carlos,
c/ Tulipán s/n Móstoles Madrid, Spain
e-mail: {juanjose.pantrigo,abraham.duarte}@urjc.es

E. Alba et al. (Eds.): Metaheuristics for Dynamic Optimization, SCI 433, pp. 291–308.
springerlink.com

the amount of research undertaken on stationary optimization problems, relatively little work has been devoted to dynamic problems, despite the potential economic advantages in doing so [8, 25]. Any advance in this field would translate to increased company profits, lower consumer prices and improved services [26].

Unlike stationary problems, dynamic problems often lack well defined optimization functions, standard benchmarks or criteria for comparing solutions [3, 8, 25, 26]. In the last decade, mainly used strategies have been specific heuristics [26] and manual procedures [2, 25]. More recently, has been proposed a metaheuristic approach. Metaheuristics are high-level general strategies for designing heuristics procedures [4]. The relevance of metaheuristics is reflected in their application for solving many different real-world complex problems, mainly combinatorial [4, 11]. Since the initial proposal of Glover about Tabu Search in 1986, many metaheuristics have emerged to design good general heuristics methods for solving different problems. The well known metaheuristics procedures are Genetic Programming, GRASP, Simulated Annealing or Ant Colony Optimization. The reader can find a review of this methods in [4, 11].

In dynamic optimization problems, metaheuristic-based approaches usually consider one of the two following strategies after events: (i) restart the search method from scratch or (ii) start from the best solutions found before the last event. In the first approach, the derived problem is processed as unrelated with respect to its origin problem. Therefore, the useful information obtained in the previous time steps is wasted, increasing the computation time. On the other hand, starting the method from the best solutions found in the previous time step could implicate a loss of diversity in the solution set. Moreover, the best solutions found in previous time steps could be close to local optima in the current one. As a consequence, the search algorithm could get stuck in a local optimum.

The filtering problem concerns about updating the present state of knowledge and predicting with drawing inferences about the future state of the system [6]. Sequential Monte Carlo population-based 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) [6]. Assuming relative small problem data changes, the new optimum location should be related to the solution of the previous problem definition. Thus, the actual search process could use previous knowledge for a more efficient search. A reasonable trade-off between the analysis of the prior problem solution and actual problem computational effort must be found. Diverse methods consisting of the low-level hybridization of Particle Filters and metaheuristics have been successfully applied to dynamic optimization problems [22–24]. The term low-level hybridization refers to the functional composition of a single optimization method. In this hybrid class of optimization methods, a given function of a metaheuristic is replaced by another metaheuristic [28]. In this work, we apply the Scatter Search Particle Filter (SSPF) for the Dynamic Travelling Salesman Problem. SSPF combines sequential estimation strategies (Particle Filter) [1, 6] and metaheuristics methods (Scatter Search) [4] in two different stages.

In the Particle Filter (PF) stage, a particle set is propagated and updated to obtain new particle sets in every time step. In the Scatter Search (SS) stage, some solutions from the particle set are selected and combined, in order to obtain better solutions. SSPF was firstly presented in [23] and it was applied to multidimensional object tracking (articulated and multiple object tracking). This work considers a dynamic variant of the Travelling Salesman Problem [9] in which distances among cities vary over the time. The problem instances used for this study have 25, 50, 75 and 100 cities placed on the Euclidean space, and a modification of approximately 15% of the previous data graph is generated as successive events for the next problem instance. Also, dynamic versions of TSPLIB benchmark are used in order to compare the performance of different approaches. Experimental results have shown that the proposed algorithm has an acceptable performance when applied to the Dynamic Travelling Salesman Problem (DTSP). Specifically, the CPU for the rest of the derived problem instances significantly decreases with respect to the initial graph problem. This reduction in the search time will not affect the quality of the solution of derived graphs.

13.2 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 [9]. The instances of this problem can be represented as a graph $G = \{V, E, W\}$, where V is a set of vertexes representing the cities, E is a set of edges which model the paths connecting cities and W is a symmetric matrix of weights that store the distances among cities. We suppose that there is an edge connecting every pair of cities. The TSP can be described as the problem of finding a Hamiltonian circuit with minimum length in the graph G [30].

The TSP has been one of the most considered problems in Combinatorial Optimization and Operations Research. This problem belongs to the NP-hard class, as demonstrated in [17]. A relevant review of different approaches used to solve this problem can be found in [12].

The Dynamic Travelling Salesman Problem (DTSP) is a generalization of the TSP in which G is time-dependent. The DTSP is general enough to be a benchmark problem. Moreover, this problem has got several practical applications such as modeling the traffic in cities along time [9] or fluctuating set of active machines [15].

Two different DTSP varieties in the literature have been described. The first one consists of inserting or deleting cities into a given problem instance [13, 14]. A different approach has been taken in [9]. They keep a constant number of cities but allow distance changing among them. The second model is applied to describe traffic jams and motorways connecting cities. In this context, before the traffic jam has occurred, good old solutions may not be optimal and the salesman needs to be re-routed. In this work, we focus on this second approach.

Ant Systems (AS) and Evolutionary Computation (EC) have been the most common metaheuristics employed for solving the DTSP. In [9, 14, 15, 26] different AS implementations were applied to DTSP. The main reason for using AS to dynamic problems is based on the pheromone concept. Pheromone can be exploited as positive and negative reinforcement and, therefore, as a way for transferring knowledge. When a change is detected in the problem instance, a partial decomposition-reconstruction procedure is performed over old solutions [26]. This process determines which elements of ant's solutions must be discarded in order to satisfy the feasibility of the new conditions.

Evolutionary Computation has been successfully applied to several combinatorial problems, including TSP. There have many algorithms based on EC to solve static TSP that can be directly applied to DTSP [30]. EC is considered to be one of the best algorithms for solving DTSP, taking into account the characteristics such that relative large population, statistical convergence, global optimization and considerable robustness [30, 31]. Generally, these implementations are based on the Inver-Over operator [16], one of the fastest algorithms in solving TSP [16, 30].

13.3 Sequential Estimation Algorithm: Particle Filter

Many interesting problems in science and engineering require estimation of the state of a system that changes over time using a sequence of noisy measurements made on the system [1]. Tracking problems, which consist of the estimation of the position of one or multiple targets moving in a scenario along time [20], are important examples of sequential estimation problems. The state-space modelling of these systems focuses on the state vector, which contains all relevant information required to describe the system under investigation. In tracking problems, for example, this information describes kinematic characteristics of the target as position, orientation, velocity, etc.

A particle filter (PF) is based on a large population of discrete representations (called particles) of the probability density function (*pdf*) which describes the evolution of a given system [6]. Particle Filters are algorithms in which theoretical distributions in the state-space are approximated by simulated random measures (also called particles) [6]. The state-space model consists of two processes: (i) an observation process $p(Z_{1:t}|X_t)$, where X_t denotes the system state vector and Z_t is the observation vector at time t, and (ii) a transition process $p(X_t|X_{t-1})$. Assuming that observations $\{Z_0, Z_1, \ldots, Z_t\}$ are sequentially measured in time, the goal is the estimation of the new system state at each time step. In the framework of Sequential Bayesian Modeling, the posterior *pdf* is estimated in two stages:

(a) Evaluation: the posterior *pdf* $p(X_t|Z_{1:t})$ is computed using the observation vector $Z_{1:t}$:

$$p(X_t|Z_{1:t}) = \frac{p(Z_t|X_t)p(X_t|Z_{1:t-1})}{p(Z_t|Z_{1:t-1})} \tag{13.1}$$

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

$$p(X_t|Z_{1:t-1}) = \int p(X_t|X_{t-1})p(X_{t-1}|Z_{1:t-1})dX_{t-1}. \tag{13.2}$$

The aim of the PF algorithm is to recursively estimate the posterior *pdf* $p(X_t|Z_{1:t})$. This *pdf* is represented by a set of weighted particles $\{(\mathbf{x}_t^0, \pi_t^0), \ldots, (\mathbf{x}_t^N, \pi_t^N)\}$, where the weights $\pi_t^i \propto p(Z_{1:t}|X_t = \mathbf{x}_t^i)$ are normalized.

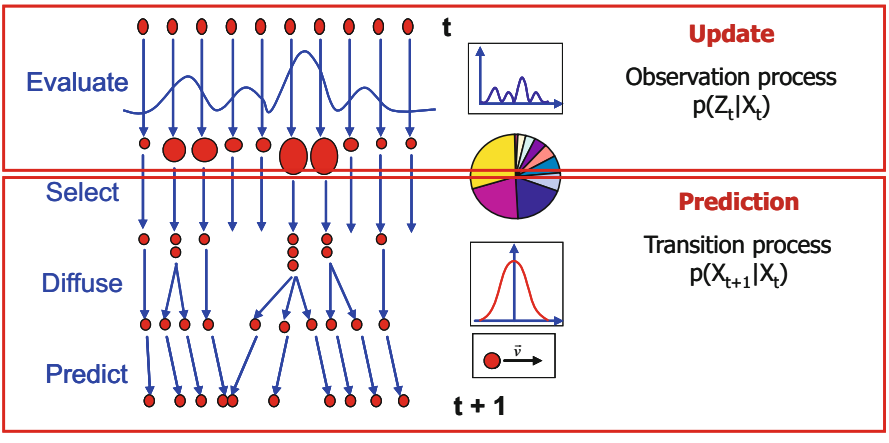

Fig. 13.1 Particle Filter scheme.

Figure 13.1 represents a schema for the PF algorithm. From an algorithmic point of view, PF directs the temporal evolution of a particle set. Particles in PF evolve according to the system model and they are selected or eliminated with a probability which depends on their weight, determined by the *pdf* [6]. In visual tracking problems this *pdf* represents the probability of a target being in a given position in the image. As a consequence, the utility of the particle filter algorithm for dynamic opimization problems lies in the description of the temporal evolution of the system state.

Therefore, Particle Filters can be seen as algorithms handling the particles time evolution. Particles in PF move according to the state model and are multiplied or died according to their weights or fitness values as determined by the likelihood function [6].

13.4 Population Based Metaheuristic: Scatter Search

Scatter Search (SS) [11, 18] 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 (never randomly) 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 relevant review of this method can be found in [18].

In Figure 13.2 an outline of the SS is shown. SS procedure starts by choosing a subset (called $RefSet$) from a solution set S of $PopSize = |S|$ initial feasible ones. The solutions in $RefSet$ are obtained by choosing the h best solutions and the r most diverse ones in S. Then, new solutions are generated by making combinations of solution subsets (pairs typically) from $RefSet$. The goal of the combination method is to produce new better solutions using information from solutions in the $RefSet$. 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 $RefSet$ quality.

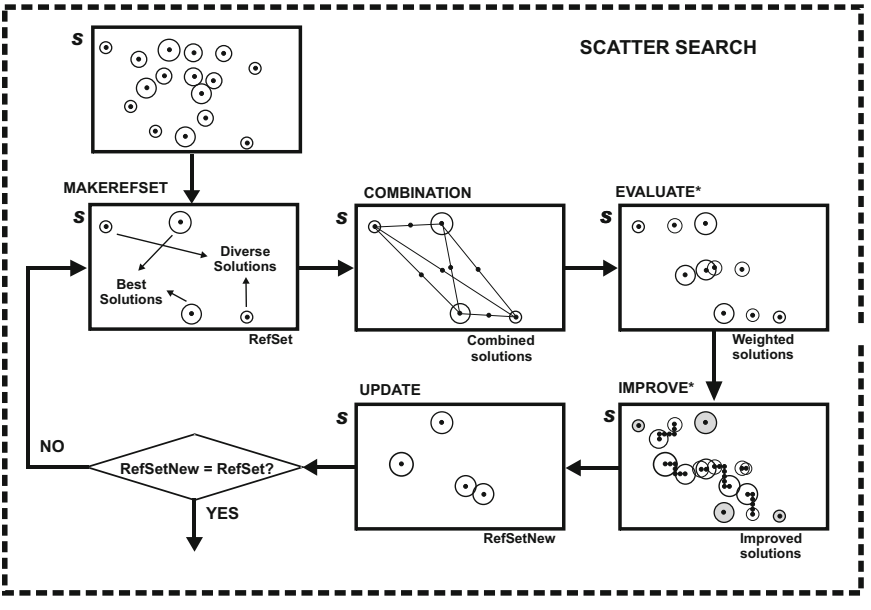

Fig. 13.2 Scatter Search scheme.

13.5 Scatter Search Particle Filter

Dynamic optimization problems deals with optimization techniques, but also with prediction tasks. This assumption is supported by 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 events in order to find high quality solutions for new derived problem 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 (trajectorial metaheuristic) or a set of last solutions (population based metaheuristic) 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, use 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. In addition, the $RefSet$ could be improved with diverse solutions.

Making a decision of what information is propagated to the next time steps is very important. 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. 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, from our viewpoint dynamic optimization problems needs 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 order to ask this question, a novel hybrid algorithm called *Scatter Search Particle Filter* (SSPF) is proposed to solve the Dynamic TSP.

13.5.1 Scatter Search and Particle Filter Hybridization

SSPF hybridizes both Scatter Search (SS) and Particle Filter (PF) frameworks in two different stages:

- In the *Particle Filter stage*, a particle (solution) 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 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.

Figure 13.3 shows the hybridization of SS and PF algorithms to obtain the SSPF algorithm. As stated in previous sections, PF algorithm can be factorized in *prediction*

and *update* stages. SS optimization is performed between these two stages. PF connection with SS is achieved by means of the selection procedure. Solutions found by SS are incorporated into the particle set of PF using the inclusion procedure.

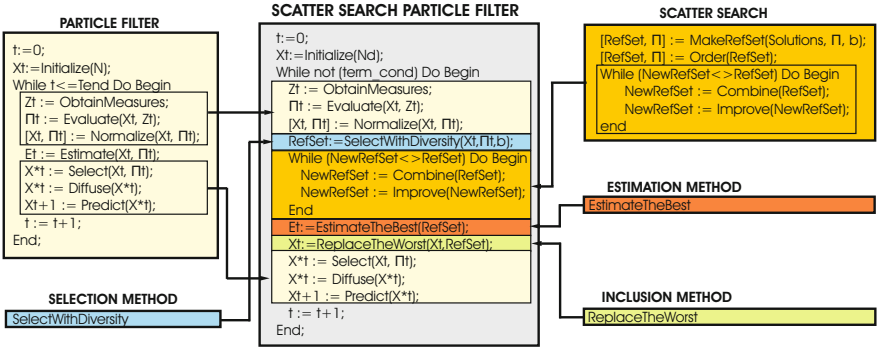

Fig. 13.3 SSPF construction starting from SS and PF.

Figure 13.4 depicts a graphical template of the SSPF algorithm. Dashed lines separate the two main components in the SSPF scheme: PF and SS optimization, respectively. SSPF starts with an initial population of N particles drawn from a known pdf (Figure 13.4: INITIALIZE stage). Each particle represents a possible solution of the problem. Particle weights are computed using a weighting function (Figure 13.4: EVALUATE stage). 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 13.4: MAKEREFSET stage). 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 13.4: COMBINE and EVALUATE stages). In order to improve the solution fitness, a local search from each new solution is performed (Figure 13.4: IMPROVE stage). Worst solutions in the *RefSet* are replaced when there are better ones (Figure 13.4: UPDATEREFSET stage). SS stage ends when new generated solutions *NewRefSet* 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 *NewRefSet* solutions (Figure 13.4: INCLUDE stage). Then, a new population of particles is created by selecting the individuals from particle set with probabilities according to their weights (Figure 13.4: SELECT and DIFFUSE stages). Finally, particles are projected into the next time step by following the update rule (Figure 13.4: PREDICT stage).

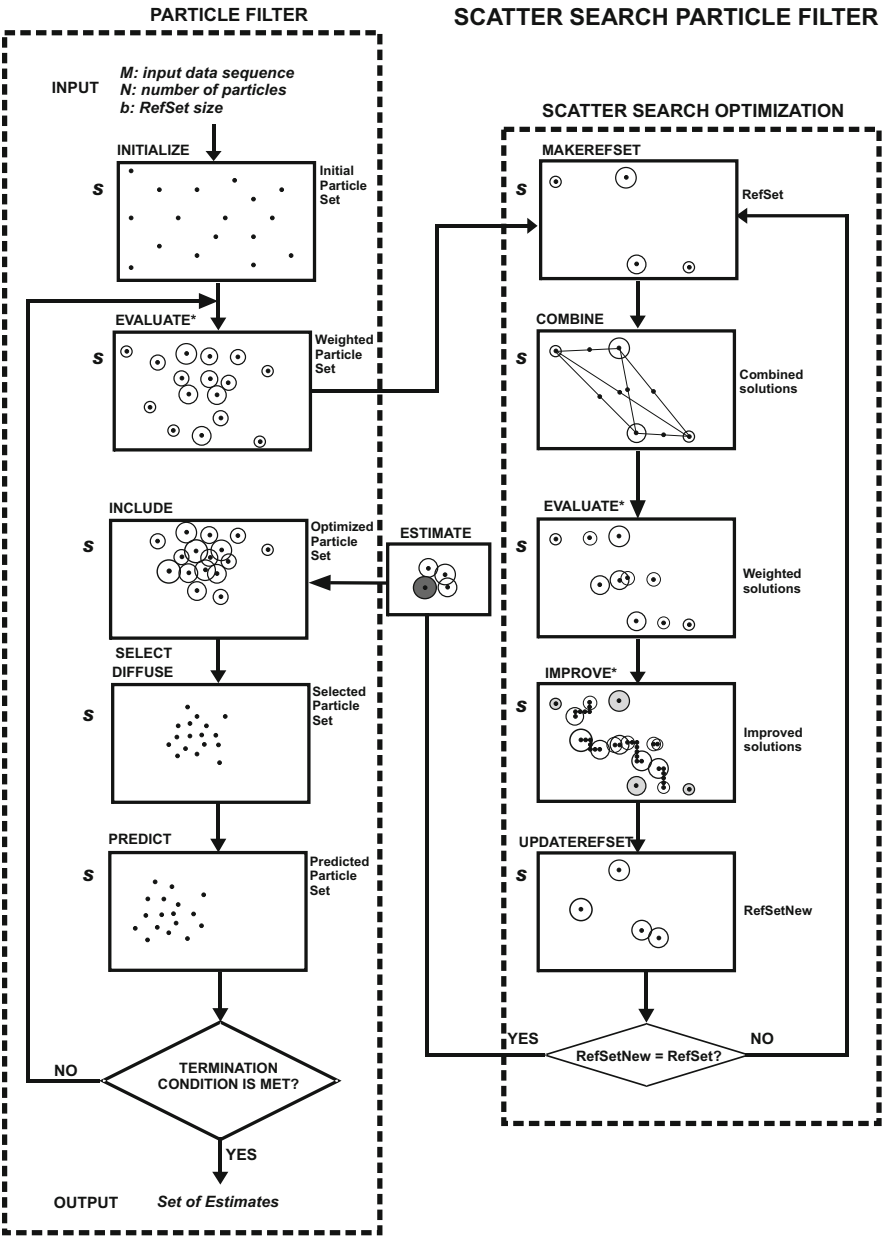

Fig. 13.4 Scatter Search Particle Filter scheme. Weight computation is required during EVALUATE and IMPROVE stages (*).

13.5.2 Scatter Search Particle Filter Main Features

The aim of the SSPF is to lead 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 for the next event $S(t+1)$, PF performs two tasks over the set $S(t)$: select the best solutions in t and predict their most probable location in the event $t+1$. Firstly, the selection procedure selects particles in a weighted random procedure, in such a way that the larger the weight of a particle, the larger the probability to select it. Secondly, PF performs a prediction procedure over the selected solutions to obtain the set $S(t+1)$. As results, we expect that solutions in $S(t+1)$ will be closer to global optimum than other solutions obtained randomly. As a complement, PF performs a diffusion procedure to the selected solutions to preserve the needed diversity in the set $S(t+1)$. In this way, solutions to be included in the *RefSet* in the time $t+1$ will be selected from a set of better solutions than a randomly obtained set. This is the main reason why SSPF reduces the required number of evaluations for the fitness function, and hence the computational load. 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 an 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. There should be enough particles to support a set of diverse solutions, avoiding the loss of diversity in the particle set. Therefore, N influences 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 $b = 10$ [5, 18].
- 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 A 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 $RefSet$.

In this research field it is usual to perform a preliminary experimentation to achieve the parameter setting.

13.6 Applying SSPF to Solve the DTSP

The main details of the SSPF implementation to solve the dynamic TSP are described in this section. The parameter setting as well as the combination and the improvement methods are detailed.

In our implementation of the SSPF, 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. Specifically, N varies from 100 for the 25-cities problem instances to 1000 for the 100-cities problem instances. The $RefSet$ is created by selecting the 5 best solutions and the 5 most diverse ones in S, according to the scatter search algorithm.

In order to find the most diverse solutions, the distance metric for *R-permutation problems* was used [18]. DTSP is considered as an R-permutation problem [18]. In these problems, relative positioning of the elements is more important than absolute positioning. As a result, the distance between two permutations p and q for R-permutation problems is defined as:

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

Voting method [18] has been used as the 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. An example can be seen in Figure 13.5.

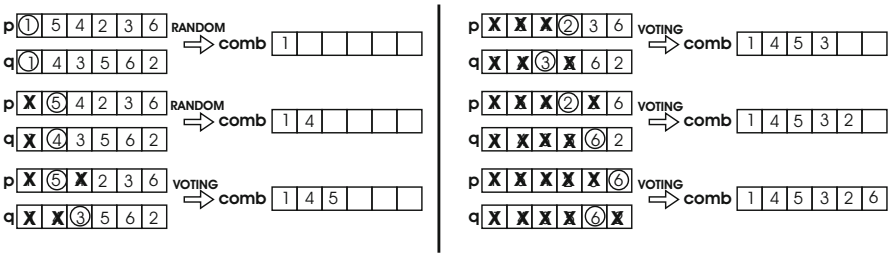

Fig. 13.5 Voting Method.

The *2-opt method* [21, 29] is usually employed as improvement stage in the SS scheme (Figure 13.4). Given a solution, consider all pairs of edges connecting four different cities. Remove two edges from the solution tour. Then there is a unique way of reconnecting the two remaining paths in such a way 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 improvement is produced.

13.7 Experimental Results

Our experimental design considers the performance comparison of SSPF with respect to the methods Scatter Search and Evolutionary Algorithms. The computational

experiments were conducted in an Intel Pentium architecture. All algorithms were coded in MATLAB 6.1, without optimization and by the same programmer to have comparable results. Different implementations are applied to several instances of DTSP and results are compared. The following sections are devoted to describe the considered problem instances, the implemented algorithms and the obtained results.

13.7.1 Problem Instances

Unfortunately, as far as the authors know, there are no benchmarks for the DTSP. Thus, we generated two sets of instances, called as synthetic and standard-based data. Synthetic data are composed by four different graph sequences. They were created, using 25, 50, 75 and 100 cities. Each sequence is composed by 10 different derived graphs. In order to know the value of the optimum, cities are located in the Euclidean space, along the diagonal as shown in figure 13.6. In the first frame, cities are located in lexicographic order along the diagonal. Subsequent frames are generated by performing exchanges of cities in groups of three as shown in figure 13.6. The average probability of node exchange in the graph sequence is $p_{change} = 0.15$.

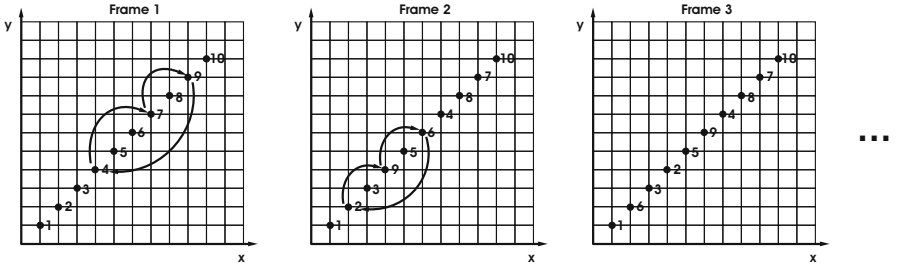

Fig. 13.6 Graph sequence generation process for synthetic instances.

Standard-based data are built as dynamic version of benchmarks from the public-domain library TSPLIB [27]. Specifically, these instances are dynamic modifications of BAYG29, BERLIN51 and ST70. These graphs belong to Euclidean symmetric class. 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. Figure 13.7 shows the sequence derived from BERLIN51.

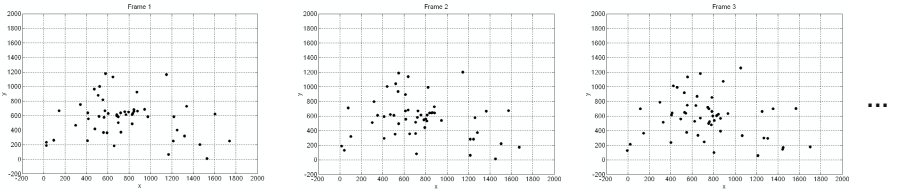

Fig. 13.7 Graph sequence generation process for standard-based instances. The first frame corresponds with the original TSPLIB instance

13.7.2 Implementation Details of the Considered Algorithms

We have implemented different versions of the Scatter Search and Evolutionary Algorithms in order to compare their performance with respect to the SSPF. The main details are described in this section. All procedures use as stopping criteria *10 million objective function evaluations or no improvement in the mean population fitness*. Solutions are coded as tours connecting cities.

13.7.2.1 Scatter Search Implementation

We have developed two different Scatter Search implementations. The first one, called SS1, considers each graph in the sequence is totally decoupled one from the others. Therefore, computation is restarted from scratch after events. In the second implementation (called SS2) the problems are supposed to be quite related, so the $RefSet$ obtained in the previous time step is used as a new $RefSet$ for the next one. SS2 does not use auxiliary methods to add diversity in the $RefSet$.

SS parameters $PopSize$ and b for both implementations SS1 and SS2 were set to 100 and 10, respectively, as recommended in [5]. In order to obtain comparable results, we use the same $RefSet$ composition, combination and improvement methods as in SSPF implementation.

13.7.2.2 Evolutionary Algorithm Implementation

The SSPF algorithm is also compared with an Evolutionary Algorithm (EA). EA performs the main stages of a standard genetic algorithm, including an improvement stage. The algorithm uses voting method as crossover operator and 2-opt method as improvement method.

EA parameters were set to $PopSize = 100$, crossover probability $p_c = 0.25$ and mutation probability $p_m = 0.01$ as recommended in [21]. Finally, improvement probability was set to $p_i = 0.25$.

13.7.3 Computational Testing

Experimental results are divided in to two sections. In the first one, we test SSPF over the synthetic instances to justify the convenience of the method. The second one is devoted to the comparison of the performance of the SSPF with respect to SS and EA.

13.7.3.1 SSPF in Synthetic Data

In this section, experimental results obtained by applying our proposal to synthetic data are shown. In table 13.1, 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 our algorithm.

Table 13.1 Average execution time values over 10 runs for each graph sequence

Number of Cities	Size of Particle Set	Average Time SS Solution	Average Time SSPF Solution	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

Figure 13.8 shows the average execution time per frame over 10 runs of the same graph sequence. Each sequence is composed by 10 similar graphs. In this figure, relative execution time is represented for each frame. As it can be observed, execution time for the 2nd to 10th graphs (SSPF improvement) is significantly smaller than the execution time for the first graph (SS approach) in all considered instances.

Results show that SSPF achieves the best solution in all instances. Moreover, it is faster than SS1 implementation without loss of quality.

13.7.3.2 SSPF vs. SS & EA in Standard-Based Data

This section presents a comparison between SSPF and the implementations of SS and EA. Results obtained by these algorithms (SS1, SS2, EA and SSPF) over all standard-based data (BAYG29, BERLIN52 and ST70) are presented in figure 13.9. Because initial conditions and initial procedures performed are the same in SS1,

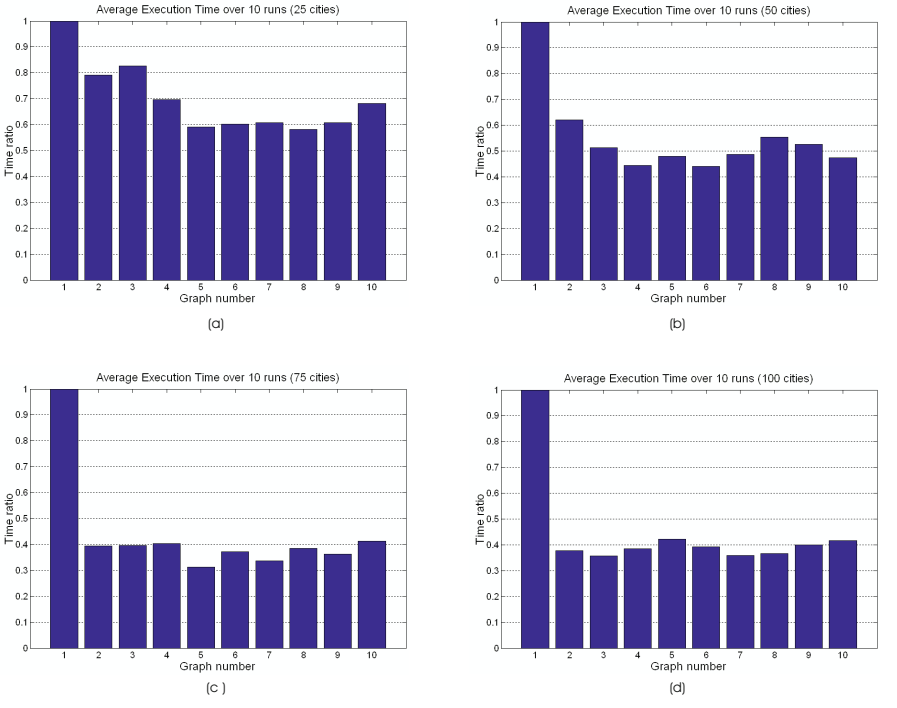

Fig. 13.8 Execution time per graph in (a) 25, (b) 50, (c) 75 and (d) 100-cities problem.

SS2 and SSPF, solutions found in the first graph are 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 dissimilar.

Quality of estimation performed by SS1 and SSPF is similar in subsequent graphs. However, execution time is significantly lower in the SSPF approach, as explained in previous sections. In the SS2 implementation. the search procedure is trapped in a local optimum (maybe in in the neighbourhood of the optimum found in the previous time step). This yields SS2 that achieves the lowest execution time, but with very poor quality. Finally, EA finds good quality solutions, but the time required to obtain it is larger than using SSPF.

In table 13.2, a resume of main results obtained using different implementations is shown. Average execution time and path lengths values demonstrate the better performance of SSPF.

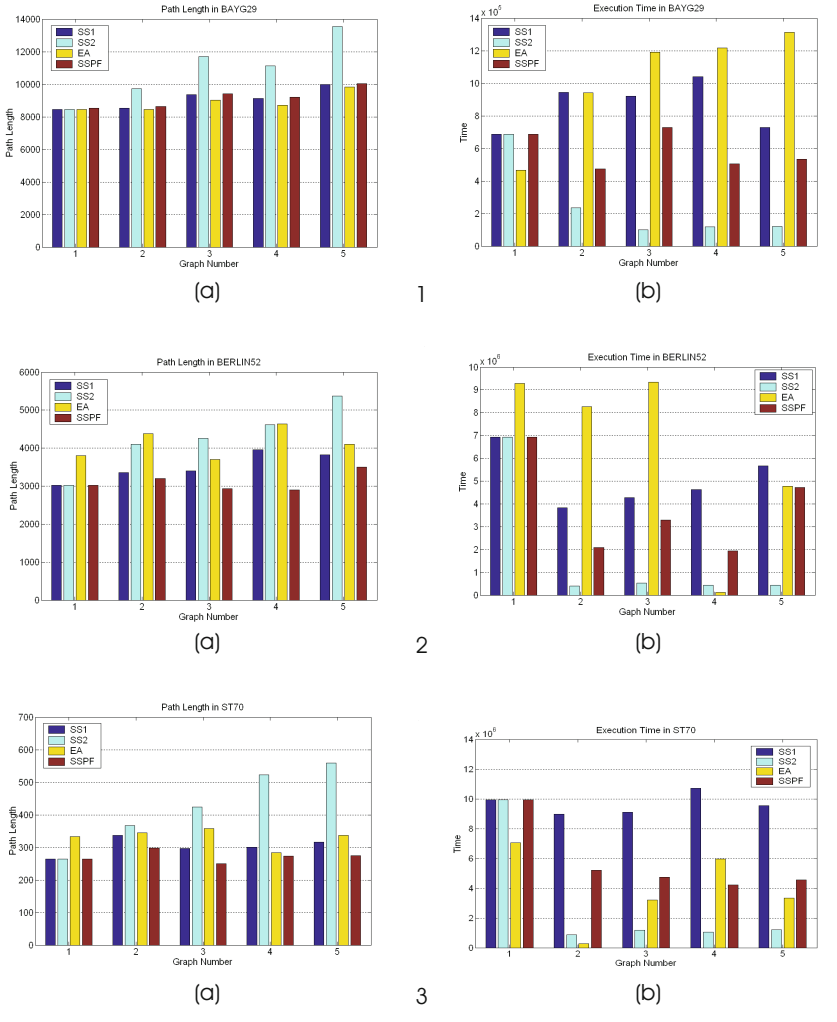

Fig. 13.9 Voting Method.

Table 13.2 Average execution time and path lengths over all instances

	SS1		**SS2**		**EA**		**SSPF**	
	Length	*Time*	*Length*	*Time*	*Length*	*Time*	*Length*	*Time*
*BAYG*29	0.86×10^6	0.91×10^4	0.25×10^6	1.09×10^4	1.02×10^6	0.89×10^4	0.59×10^6	0.91×10^4
*BERLIN*52	5.07×10^6	3.51×10^3	1.75×10^6	4.27×10^3	6.37×10^6	4.12×10^3	3.79×10^6	3.11×10^6
*ST*70	9.65×10^6	302.97	2.84×10^6	427.58	3.9×10^6	331.15	5.72×10^6	272.15

13.8 Conclusions

In this work we successfully applied the Scatter Search Particle Filter (SSPF) algorithm to the Dynamic Travelling Salesman Problem (DTSP). Experimental results show that SSPF appreciably increases the performance of Scatter Search and Evolutionary Algorithm methods in a challenging dynamic optimization problem, without losing quality in the estimation procedure. This improvement becomes even more significant for large instances.

References

[1] Arulampalam, M., et al.: A Tutorial on Particle Filter for Online Nonlinear/Non-Gaussian Bayesian Tracking. IEEE Trans. on Signal Processing 50(2), 174–188 (2002)
[2] Beasley, J., Sonander, J., Havelock, P.: Scheduling Aircraft Landings at London Heathrow using a Population Heuristic. Journal of the Operational Research Society 52, 483–493 (2001)
[3] Beasley, J., Krishnamoorthy, M., Sharaiha, Y., Abramson, D.: The displacement Problem and Dynamically Scheduling Aircraft Landings, Working paper (2002), http://graph.ms.ic.ac.uk/jeb/displace.pdfm
[4] Blum, C., Roli, A.: Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys 35(3), 268–308 (2003)
[5] Campos, V., Laguna, M., Marti, R.: Scatter Search for the Linear Ordering Problem. In: New Ideas in Optimization. McGraw-Hill (1999)
[6] 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)
[7] Dorigo, M., Gambardella, L.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IIEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
[8] Dror, M., Powell, W.: Stochastic and Dynamic Models in Transportation. Operations Research 41, 11–14 (1993)
[9] Eyckelhof, C., Snoek, M.: Ant Systems for A Dynamic DSP: Ants Caught in a Traffic Jam. In: Proc. of ANTS 2002 Conference (2002)
[10] Glover, F.: A Template for Scatter Search and Path Relinking. In: Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M., Snyers, D. (eds.) AE 1997. LNCS, vol. 1363, pp. 13–53. Springer, Heidelberg (1998)
[11] Glover, F., Kochenberger, G.: Handbook of metaheuristics. Kluwer Academic Publishers (2002)
[12] Gutin, G., Punnen, A.: The traveling salesman problem and its variations. Kluwer Academic Publishers (2004)
[13] Guntsh, M., Middendorf, M., Schmeck, H.: An Ant Colony Optimization Approach to Dynamic TSP. In: Proc. GECCO-2001 Conference, pp. 860–867. Morgan Kaufmann Publishers, San Francisco (2000)
[14] Guntsch, M., Middendorf, M.: Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)

[15] Guntsch, M., Middendorf, M.: Applying Population Based ACO to Dynamic Optimization Problems. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) Ant Algorithms 2002. LNCS, vol. 2463, pp. 111–122. Springer, Heidelberg (2002)
[16] Tao, G., Michalewicz, Z.: Inver-over Operator for the TSP. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 803–812. Springer, Heidelberg (1998)
[17] Karp, R.: Reducibility among Combinatorial Problems. In: Miller, R., Thatcher, J. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press (1972)
[18] Laguna, M., Marti, R.: Scatter Search methodology and implementations in C. Kluwer Academic Publisher (2003)
[19] Larsen, A.: The dynamic vehicle routing problem. PhD Thesis (2000)
[20] MacCormick, J.: Stochastic Algorithm for visual tracking. Springer (2002)
[21] Michalewitz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer (1996)
[22] Pantrigo, J.J., Sánchez, Á., Gianikellis, K., Duarte, A.: Path Relinking Particle Filter for Human Body Pose Estimation. In: Fred, A., Caelli, T.M., Duin, R.P.W., Campilho, A.C., de Ridder, D. (eds.) SSPR&SPR 2004. LNCS, vol. 3138, pp. 653–661. Springer, Heidelberg (2004)
[23] Pantrigo, J.J., Sánchez, A., Montemayor, A.S., Duarte, A.: Multi-Dimensional Visual Tracking Using Scatter Search Particle Filter. Pattern Recognition Letters 29(8), 1160–1174 (2008)
[24] Pantrigo, J.J., Hernández, A., Sánchez, A.: Multiple and Variable Target Visual Tracking for Video Surveillance Applications. Pattern Recognition Letters 31(12), 1577–1590 (2010)
[25] Sadeh, N., Kott, A.: Models and Techniques for Dynamic Demand-Responsive Transportation Planning. Technical Report, CMURI- TR-96-09, Robotics Institute, Carnegie Mellon University (1996)
[26] Randall, M.: Constructive Meta-heuristics for Dynamic Optimization Problems. Technical Report, School of Information Technology, Bond University (2002)
[27] Reinelt, G.: TSPLIB. University of Heidelberg (1996), `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`
[28] Talbi, E.-G.: A Taxonomy of Hybrid Metaheuristics. Journal of Heuristics 8(5), 541–564 (2002)
[29] Vizeacoumar, F.T.: Implementation. Project report Combinatorial Optimization CMPUT - 670 (2003)
[30] Zhang-Can, H., Xiao-Lin, H., Si-Duo, C.: Dynamic traveling salesman problem based on evolutionary computation. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, pp. 1283–1288 (2001)
[31] Liu, Z., Kang, L.: A Hybrid Algorithm of n-OPT and GA to Solve Dynamic TSP. In: Li, M., Sun, X.-H., Deng, Q.-n., Ni, J. (eds.) GCC 2003, Part II. LNCS, vol. 3033, pp. 1030–1033. Springer, Heidelberg (2004)