# A Set-based Comprehensive Learning Particle Swarm Optimization with Decomposition for Multiobjective Traveling Salesman Problem

Xue Yu[124], Wei-Neng Chen[1234*], Xiao-Min Hu[245], Jun Zhang[1234]
[1]School of Advanced Computing, Sun Yat-sen University, Guangzhou, China
[2]Key Lab. Machine Intelligence and Advanced Computing, Ministry of Education, China
[3]Collaborative Innovation Center of High Performance Computing, China
[4]Engineering Research Center of Supercomputing Engineering Software, Ministry of Education, China
[5]School of Public Health, Sun Yat-sen University, Guangzhou, China
* Corresponding Author, Email: chenwn3@mail.sysu.edu.cn

## ABSTRACT
This paper takes the multiobjective traveling salesman problem (MOTSP) as the representative for multiobjective combinatorial problems and develop a set-based comprehensive learning particle swarm optimization (S-CLPSO) with decomposition for solving MOTSP. The main idea is to take advantages of both the multiobjective evolutionary algorithm based on decomposition (MOEA/D) framework and our previously proposed S-CLPSO method for discrete optimization. Consistent to MOEA/D, a multiobjective problem is decomposed into a set of subproblems, each of which is represented as a weight vector and solved by a particle. Thus the objective vector of a solution or the cost vector between two cities will be transformed into real fitness to be used in S-CLPSO for the exemplar construction, the heuristic information generation and the update of *pBest*. To validate the proposed method, experiments based on TSPLIB benchmark are conducted and the results indicate that the proposed algorithm can improve the solution quality to some degree.

## Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *Heuristic methods*.

## General Terms
Algorithms, Design, Experimentation.

## Keywords
multiobjective traveling salesman problem (MOTSP); multiobjective evolutionary algorithm based on decomposition (MOEA/D); set-based particle swarm optimization (S-PSO).


*GECCO '15, July 11 - 15, 2015, Madrid, Spain*
 ISBN 978-1-4503-3472-3/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2739480.2754672

## 1. INTRODUCTION
As one of the most famous combination optimization problems [1], the traveling salesman problem (TSP) is to find a Hamiltonian circuit with minimum length in a number of cities, that is, a salesman would like start from a home city to visit other cities once and only once and come back to his home city finally. The TSP has been proved as NP-hard [2]. Because of the simplicity to describe and hardness to solve, TSPs are commonly used as benchmarks to assess the performance of heuristics [3] and have been well research in many fields [4]-[7]. However, in reality some other kinds of cost except for distance have to be considered when dealing with a TSP, such as risk and time, thus the TSP becomes an MOTSP with more than one objectives to optimize. A MOTSP can be formulated as follows

$$\begin{aligned} &\min F(x) = (f_1(x),...,f_m(x)) \\ &f_i(x) = \sum_{j=1}^{n-1} c^i_{x_j,x_{j+1}} + c^i_{x_1,x_n}, i = 1,...,m \end{aligned} \quad (1)$$

where $m$ is the number of objectives, n is the number of cities, $x = (x_1,...,x_n)$ is a permutation of city indices, $C(i) = (c^i_{j,k})_{n\times n}$ is the $i$th cost matrix, $f_i(x)$ is the $i$th objective. It is worth noting that the objectives in (1) often conflict with each other, thus a single solution with every objective optimized is impossible to find. Like other multiobjective optimization problems (MOPs), a set of tradeoff solutions is usually found when developing an MOEA for dealing with an MOTSP. A prevailing way to make tradeoff in MOPs is based on Pareto dominance theory, where a *Pareto front* (*PF*) is found in the objective space (*Pareto set* (*PS*) in the decision space) [8].

Much effort has been devoted to solving MOTSPs and there are two approaches regularly adopted: 1) based on local search on multiobjective framework; 2) based on EAs and multiobjective framework. In the former case, the hybridization of decomposition and local search is developed in [9], where the Pareto local search is extended to solve MOTSPs. For the latter case, more applications can be found. For instances, the estimation of distribution algorithm (EDA) [10] is integrated into the decomposition framework to solve the multiobjective multiple traveling salesman problem in [11]; L. J. Ke etc. [12] proposed a multiobjective EA, namely MOEA/D-ACO, combining ant colony optimization (ACO) [13] and the multiobjective evolutionary algorithm based on decomposition (MOEA/D) [14];

[15] developed a multiobjective hybrid genetic algorithm (MO-HGA) for TFT-LCD module assembly scheduling that hybridizes with the variable neighborhood descent (VND) algorithm as a local search and TOPSIS evaluation technique to derive the best compromised solution.

Among all the popular EAs, the particle swarm optimization (PSO) is easy to implement as well as obtain pretty solutions when applied to optimize many problems. However, the conventional PSO was originally designed for solving continuous problems [16] and some strategies have been proposed to develop discrete PSO when using it to solve discrete problems [17]-[20]. Among those discrete strategies, the set-based particle swarm optimization (S-PSO) method proposed in [20] is effective and its validity for solving combinatorial optimization problems (COPs) have been proved on TSP and 0/1 knapsack problem benchmarks. Based on the S-PSO method, the representation scheme is carefully designed and the position and velocity of a particle are represented as a crisp set and a set with possibilities respectively. Then according to the update formulae of a variant PSO with extra mathematical operations defined for crisp sets and sets with possibilities, the discrete PSO evolves to solve some certain COPs. Like other heuristics for solving COPs, the local heuristic information is also important to the efficiency of S-PSO and should be carefully designed based on the characteristics of the COP to be solved.

To make S-PSO available for solving MOTSPs, a prevailing multiobjective framework based on decomposition, i.e. MOEA/D, is adopted in this paper to extend S-PSO to its multiobjective version. More specifically, we obtain the discrete comprehensive learning PSO (CLPSO) [21] based on S-PSO method, namely S-CLPSO and then extend S-CLPSO to its multiobjective version using the MOEA/D, donated as MOEA/D-S-CLPSO. In MOEA/D-S-CLPSO, an MOTSP is decomposed into several single-objective TSPs represented by different weight vectors, and each subproblem is optimized by a particle in S-CLPSO method using information from its neighboring particles. The neighboring particles of a particle are determined according to the distance between weight vectors of subproblems. Actually, the weight vector of a particle is also used in exemplar construction, local heuristic information generation and update of best-so-far solutions in S-CLPSO: 1) when constructing an exemplar for a particle, its weight vector is used to converted the objective vectors of two randomly-selected particles into two real values and the better one can be selected out by comparing the two real values during tournament selection. 2) The heuristic information will help to make decision when there are two cities can be taken as next node. However, there will be a cost vector instead of a single cost value between two cities. Through the product of a cost vector and the particle's weight vector, the cost vector is transformed into a single cost value and the comparison can be made easily on it. 3) In MOEA/D-S-CLPSO, the objective vectors of a particle's all neighboring particles are compared to update the particle's *pBest*, where the objective vector is transformed into a real value. To validate the proposed method, experiments based on TSPLIB benchmark are conducted and the results indicate that the proposed algorithm can improve the solution quality to some degree.

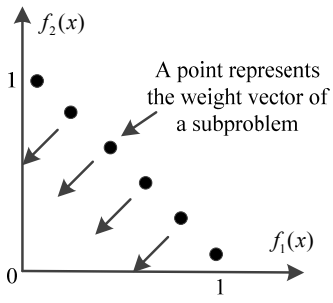



Figure 1. Example of a set of evenly distributed subproblem weight vectors

The rest of this paper is organized as follows. Section 2 reviews the MOEA/D framework. Section 3 introduces S-PSO method. In Section 4, the proposed MOEAD-S-CLPSO for solving MOTSP is presented. Experiments and comparison studies are shown in Section 5, and the conclusions are drawn in Section 6.

# 2. MOEA/D FREAMWORK

MOEA/D is a prevalent EA framework based on decomposition to solve MOPs. It decomposes an MOP into a number of scalar optimization subproblems and optimizes them simultaneously. Each subproblem is represented by a weight vector and optimized by combining the information from its several neighboring subproblems.

## 2.1 Decomposition Approaches

Several approaches are available to convert an MOP into a number of scalar optimization problems. The following two approaches are introduced in [8]:

1) *Weighted Sum Approach*

Considering a minimal MOP with m objectives, let $\lambda = (\lambda_1,...,\lambda_m)^T$ be a weight vector. Then the scalar optimization problem is defined as follows:

$$\text{minimize } g^{ws}(x \mid \lambda) = \sum_{i=1}^{m} \lambda_i f_i(x), x \in \Omega \qquad (2)$$

2) *Chebyshev Approach*

In this approach, the scalar optimization problem is defined as follows:

$$\text{minimize } g^{te}(x \mid \lambda, z^*) = \max_{1 \le i \le m} \{\lambda_i \mid f_i(x) - z_i^* \mid\}, x \in \Omega \qquad (3)$$

where $z^*$ is the reference point, i.e., $z_i^* = \min\{f_i(x) \mid x \in \Omega\}$.

## 2.2 Basic Ideas of MOEA/D

Actually, the following two main points in MOEA/D make it efficient as well as simple.

1) *Subproblem*: Each subproblem $i$ is assigned with a specific weight vector $\lambda_i$, thus it's responsible for optimizing that direction of the MOP. Therefore, a set of well-distributed subproblem weight vectors will guarantee a good approximation of the *PF*. Usually, a set of evenly distributed subproblem weight vectors is used, that is, shown in Fig. 1., where $\lambda_i^j \ge 0$ for all $j = 1,...,m$ and $\sum_{j=1}^{m} \lambda_i^j = 1$.

2) *Neighborhood*: Each subproblem is optimized by only using its neighboring subproblems' information, which results in a lower computational complexity. In MOEA/D, the neighborhood $B(i)$ of subproblem $i$ with the weight vector $\lambda_i$ is defined as a set of its

several closest weight vectors in all the subproblem weight vectors, i.e., $\{\lambda_1,...,\lambda_N\}$ .

# 3. SET-BASED PSO

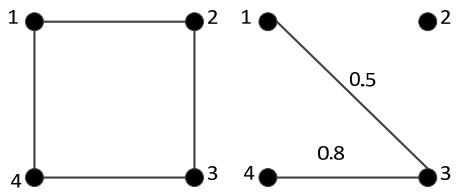
## 3.1 Continuous PSO

In the original PSO **Error! Reference source not found.**, the *i*th particle in the swarm maintains a velocity vector $v_i$ and a position vector $x_i$ , where $v_i, x_i \in R^n$ if the swarm is to search the global optimum in the *n*-dimensional space. In each iteration, the velocity and position of particle *i* are updated as follows:

$$v_i^j = wv_i^j + c_1 r_1^j (pBest_i^j - x_i^j) + c_2 r_2^j (gBest^j - x_i^j) \quad (4)$$

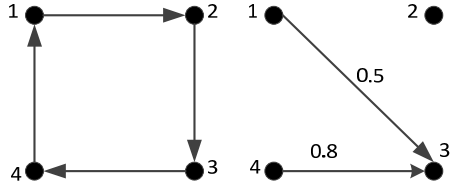
$$x_i^j = x_i^j + v_i^j \quad (5)$$

where $pBest_i$ is the best-so-far solution yielded by particle *i*, *gBest* is the best-so-far solution obtained by the whole swarm, $w, c_1, c_2$ are parameters with great importance on the algorithm efficiency.




$X = \{(1,2),(2,3),(3,4),(1,4)\}$
$X^1 = \{(1,2),(1,4)\}, X^2 = \{(1,2),(2,3)\}$
$X^3 = \{(2,3),(3,4)\}, X^4 = \{(3,4),(1,4)\}$
$V = \{(1,3)/0.5,(3,4)/0.8\}$
$V^1 = \{(1,3)/0.5\}, V^2 = \{\}, V^4 = \{(3,4)/0.8\}$
$V^3 = \{(1,3)/0.5,(3,4)/0.8\}$

(a)




$X = \{(1,2),(2,3),(3,4),(4,1)\}$
$X^1 = \{(1,2)\}, X^2 = \{(2,3)\}$
$X^3 = \{(3,4)\}, X^4 = \{(4,1)\}$
$V = \{(1,3)/0.5,(4,3)/0.8\}$
$V^1 = \{(1,3)/0.5\}, V^2 = \{\}$
$V^3 = \{\}, V^4 = \{(4,3)/0.8\}$

(b)

Figure 2. Examples of the representation scheme for the TSP. (a) Scheme for the symmetric TSP. (b) Scheme for the asymmetric TSP.

Many variants of PSO have been proposed by researchers, and CLPSO is a representative one among them. In CLPSO, a novel learning strategy is used and the velocity of particle *i* is updated as follows:

$$v_i^j = wv_i^j + cr^j (pBest_{f_i(j)}^j - x_i^j) \quad (6)$$

where $f_i(j)$ is set as *i* or the better one of two randomly selected particles in a particularly probabilistic way. In other words, all particles' historical best information is used to update a particle's velocity [21].

## 3.2 S-PSO

To extend continuous PSOs to their discrete versions, the set-based PSO method is presented in [20]. Since a set-based representation scheme is applied in S-PSO, a particle's position *X* is redefined as a crisp set corresponding to a feasible solution to the problem and its velocity *V* is redefined as a set with possibilities, also, the updating operators of position and velocity are changed respectively.

To address the key concepts in S-SPO, a TSP instance with four nodes, indexed as 1,…,4, is taken as an example in the following.

1) *Representation scheme of X and V*: Usually, the set-based representation scheme of a particle's position and velocity for a discrete problem is intuitive to build up, whose design depends on the characteristics of a specific problem. To describe it figuratively, representations of *X* and *V* for the TSP is given in Figure 2.

2) *The velocity updating rules*: The concepts of a set with possibilities, the multiplication operator between a coefficient and a set with possibilities, the multiplication operator between a coefficient and a crisp set and the plus operator between two sets with possibilities are carefully defined in [20]. Based on those definitions, examples of all the arithmetic operators when updating particle *i*'s for a symmetric TSP are given in Figure 3.

3) *Position updating*: After updating the velocity, particle *i* uses the new velocity $V_i$ , its own position $X_i$ to build a new position $NEWX_i$ . Unlike the continuous cases, the positions in the discrete space must satisfy some constraints, for example, a feasible solution in TSP must be a permutation of all the city nodes. To ensure the feasibility of the new position, a constructive is used in S-PSO with extra constraint-checking mechanism or solution-repair mechanism. When constructing the *j*th dimension of $NEWX_i$ , denoted as $NEWX_i^j$ , particle *i* first learns from elements in $cut_\alpha(V_i^j)$ , which is a set consists of all elements in $V_i^j$ whose possibility is larger than a given threshold $\alpha \in [0,1]$ . If there is no available element in $cut_\alpha(V_i^j)$ , particle *i* reuses the elements in the previous $X_i^j$ . If there is still no available element in $X_i^j$ , it uses the other available elements to complete $NEWX_i^j$ .

# 4. MOEA/D-S-CLPSO FOR MTSP

Consistent with the MOEA/D framework, MOEA/D-S-CLPSO decomposes a multiobjective optimization problem into a number of single-objective subproblems. Each particle with a certain representative weight vector is responsible for solving one subproblem, and it belongs to a neighborhood. Checking all the newly constructed solutions in the neighborhood, a particle updates its best-so-far *pBest* if a better one is found in terms of its

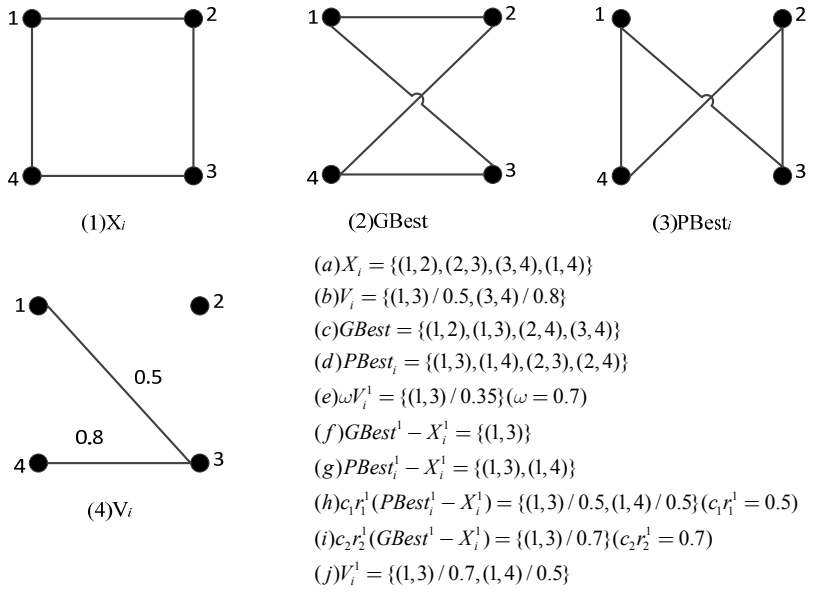



Figure 3. Examples of all the arithmetic operators when updating particle *i*'s for a symmetric TSP.

own objective. Meanwhile, exemplars for guiding each particle's evolution are constructed in CLPSO way, i.e., each dimension of a particle learns from itself or the better one from two particles randomly selected in the neighborhood according to the specific objective. A particle updates its velocity and position in the similar way in S-PSO except that the local heuristic information is closely related to the corresponding subproblem.

## 4.1 Initialization

Since the proposed MOEA/D-S-CLPSO consists of MOEA/D framework and S-CLPSO algorithm, the initialization preparations should be conducted from that two perspectives. For MOEA/D, the way of decomposing an MTSP into several subproblems (single-objective TSPs) and determining the neighboring subproblems of each subproblem should be defined, that is, the setting of weight vectors $\lambda s$ and the partition of neighborhoods according to $\lambda s$ . For S-CLPSO, the initialization of velocities and positions is important since the representation scheme of S-PSO changes along with the specific discrete problem.

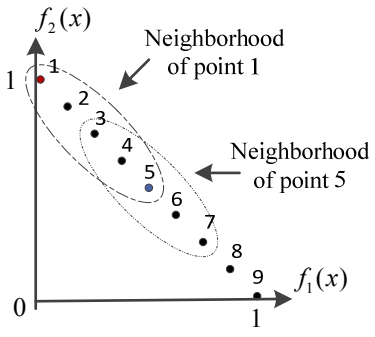



Figure 4. The neighborhoods of different points in if evenly-distributed

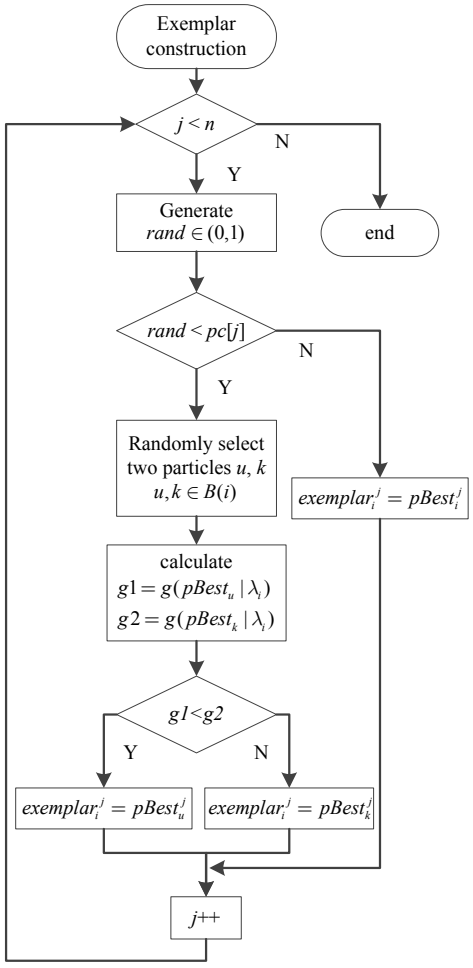



Figure 5. The algorithm of exemplar construction

1) *λs and neighborhoods*: In this paper, the evenly-distributed-like weight vectors are adopted and the neighborhoods are determined based the distances among weight vectors, that is, the neighborhood $B(i)$ of subproblem $i$ with the weight vector $\lambda_i$ is defined as a set of its several closest weight vectors in all the subproblem weight vectors, i.e., $\{\lambda_1,...,\lambda_N\}$. As for the setting of evenly-distributed-like weight vectors, more weight vectors are placed in the fringe compared with evenly-distributed weight vectors. In mathematics, take bi-objective weight vectors for example, there are $F$ weight vectors in the form of $[1,0]^T$ and $F$ weight vectors in the form of $[0,1]^T$ and the other ($ps$-2*$F$) weight vectors are evenly distributed in the way shown in Figure 1.

**Why not just evenly-distributed weight vectors**: As shown in Figure 4, assume the size of neighborhood is 5 and the number of weight vectors is 9, the neighboring points of point 1 all lay on its right except itself and the point 5 is its farthest neighbor, however, the neighboring points of point 5 lay on its left and right and the point 3 is one of its farthest neighbors. Then it is obvious that the distance between point 1 and point 5 is two times the distance between point 5 and point 3. If the distance between two weight vectors is too long, the subproblems represented by that two weight vectors will differ significantly and they should not learn from each other. Therefore, by placing more points in the fringe, the above problem can be overcome to some degree.

2) *Initialization of position and velocity*: For a TSP with $n$ cities, each position of particle $i$ is initialized as a permutation of $n$ cities, i.e. a permutation of 1,...,$n$, since the position of a particle must be a candidate solution for that TSP. Then a position {(1,3),(3,2),(2,4),(1,4)} with $n$ = 4 will represents a path that one starts from city 1, then go to city 3, then city 2, then city 4, then returns to the city 1. As for the velocity for particle $i$, we choose $n$ edges from the complete topology of $n$ cities and randomly generate a real value in [0,1] for each edge since the velocity in S-PSO is a set with possibilities. For example, a velocity can be {(1,2)/0.4, (1,3)/0.5, (2,4)/0.4, (2,3)/0.1}.

## 4.2 Exemplars construction in CLPSO way

As pointed out above, the dimension $j$ of exemplar for particle $i$ can be determined by $k$-tournament selection ($k$=2 in this paper). However, when dealing with multiobjective TSPs, the selection operator should be carefully designed due to the lack of single fitness value. Two obvious approaches can be used to select the better one: 1) based on dominance; 2) based on aggregation of multiobjective values. In the dominance-based way, it is common to find that the two randomly chosen particles not dominated by each other. Thus in this paper, the second approach has been adopted and the algorithm of exemplar construction for particle $i$ is given in Figure 5.

## 4.3 Update operators

1) *Update of velocity*: The dimension $j$ of particle $i$ is updated in formula (7). Just as the example in the section VI. A, the $V_i^1 = \{(1,2)/0.4,(1,3)/0.5\}$ and $X_i^1 = \{(1,3),(1,4)\}$, assume that w=0.7, $cr^1 = 0.7$ an $exemplar_i = \{(1,4),(4,3),(3,2),(1,2)\}$, i.e., $exemplar_i^1 = \{(1,4),(1,2)\}$, then according to the mathematical operations on crisp set and set of possibilities, the dimension $j$ of new velocity is that $V_i^1 = \{(1,2)/0.7,(1,3)/0.35\}$.

$$v_i^j = wv_i^j + cr^j(exemplar_i^j - x_i^j) \tag{7}$$

2) *Heuristic information and update of position*: When solving discrete combinatorial problems, the heuristic information is important and the bad use of heuristic information will result in terrible algorithm efficiency. For particle $i$, if the current node is city $k$ and there are two cities $u$, $l$ can be chosen as next node, then the better one should be selected out based on the heuristic information. In this paper, the multiple costs between two cities are aggregated into a real value (heuristic fitness) by the weight vector of particle $i$, i.e., the heuristic fitness from city $k$ to city $u$ is calculated as $HF_{k,u} = g(C_{k,u} \mid \lambda_i)$ where $C_{k,u}$ is costs vector from $k$ to $u$, then the heuristic fitness is used as heuristic information the one with less heuristic fitness is better. The update of X is just in the S-PSO way, that is, when constructing the $j$th dimension of $NEWX_i$, denoted as $NEWX_i^j$, particle $i$ first learns from elements in $cut_\alpha(V_i^j)$ firstly, then the elements in the previous $X_i^j$, then the other available elements. Whenever there are more than one elements available, the best one will be chosen according to their heuristic fitness values.

3) *Update of pBest*: According to the MOEA/D framework, the best-so-far solution of a particle should be updated by combining the information of its neighborhood. In MOEA/D-S-CLPSO, to update particle $i$'s *pBest*, the best one of all new solutions generated by $i$'s neighbors (including $i$ itself). Still, the aggregation method is used for selection operator, that is, the one with minimal $g$ value is selected where the $g$ value is calculated based on the decomposition approach adopted in MOEA/D, i.e., $g(X_k) = g(X_k \mid \lambda_i), k \in B(i)$.

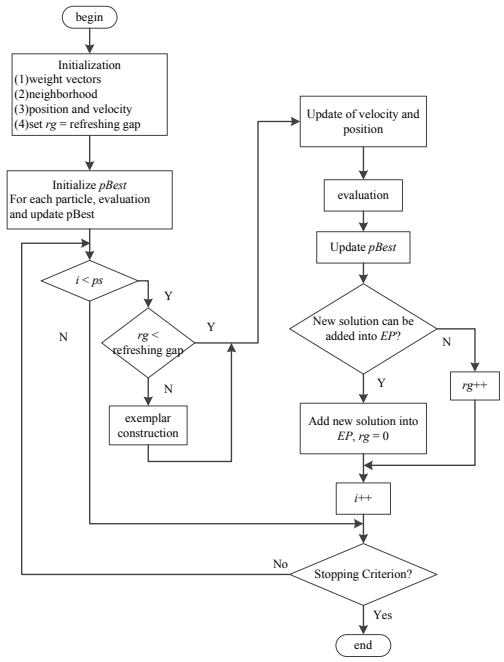



Figure. 6 The framework of MOEA/D-S-CLPSO

4) *Update of EP*: The External Pool (*EP*) is an external archive for saving all non-dominated solutions during the evolution. Whenever a new solution is generated by a particle, we check the dominance relationship between that solution and all the solutions in *EP*. If the new solution is not dominated by any solution in *EP*, then it is added into EP and all the solutions in *EP* dominated by

it will be deleted. Meanwhile, in the proposed MOEA/D-S-CLPSO, the refreshing value *rg* (used in CLPSO) of each particle will be updated according to the update of *EP*, that is, if the new solution generated by particle *i* is added into *EP* successfully then particle *i*'s refreshing value will be set to zero, otherwise, the value plus one. If the value is greater than or equals the refreshing gap, a new exemplar will be constructed for particle *i*.

## 4.4 Algorithm Framework

The main framework of MOEA/D-S-CLPSO is given in Figure 6, firstly we initialize weight vectors, neighborhoods, position and velocity for each particle and set *rg* value for refreshing gap for each particle, and then we evaluate each particle and initialize its *pBest*. After initialization, an iteration begins, that is, for each particle: construct exemplar if needed, then update velocity and position, evaluation, update *pBest* and *EP*. At the end of an iteration, the stopping criterion is checked, if stopping criterion is satisfied then the algorithm ends, otherwise a new iteration is started.

# 5. COMPARISON WITH MOEA/D-ACO

BicriterionAnt [22] is one of the best existing multiobjective ACOs on bi-objective TSP [23] [24], and MOEA/D-ACO has been proved to outperform BicriterionAnt significantly in [12]. Therefore, to assess the performance proposed in this paper, only comparison with MOEAD-ACO is made here.

## 5.1 Performance Metric

The inverted generational distance (IGD) is used to assess the performance of the algorithm proposed in this paper.

Let $P$ be the *PF* approximated by an MOEA and $P^*$ be a set of uniformly distributed points along the real *PF* of an MTSP, the IGD from $P^*$ to $P$ is defined as

$$IGD(P^*, P) = \frac{\sum_{v \in P^*} d(v, P)}{|P^*|} \tag{8}$$

where $d(v, P)$ is the minimum Euclidean distance between $v$ and the points in $P$. If $P^*$ is large enough to approximate the real *PF*, the IGD value can measure the distance from $P$ to the real *PF* to some degree. To have a low IGD value, points in $P$ must be close to the real *PF* and well distributed in objective space without missing any part of the whole *PF*.

## 5.2 Experimental Setup

The 12 instances with corresponding $P^*$s from [25] are used in the experiments. The two algorithms are implemented in C++ and executed in the same environment (the source code of MOEA/D-ACO for TSP is provided by its authors and available in [26]).

The parameters of MOEA/D-ACO are set according to **Error! Reference source not found.**. Since MOEA/D-ACO with Chebyshev decomposition strategy performs better that with Weighted Sum strategy, only Chebyshev strategy for MOEA/D-ACO is included in experiments. In contrast, the Weighted Sum strategy is used for MOEA/D-S-CLPSO and the parameter settings of MOEA/D-ACO are as follows.

- Number of particles *ps* = 72
- Neighborhood size *T* = 12
- Max generations *maxgen* = 1000

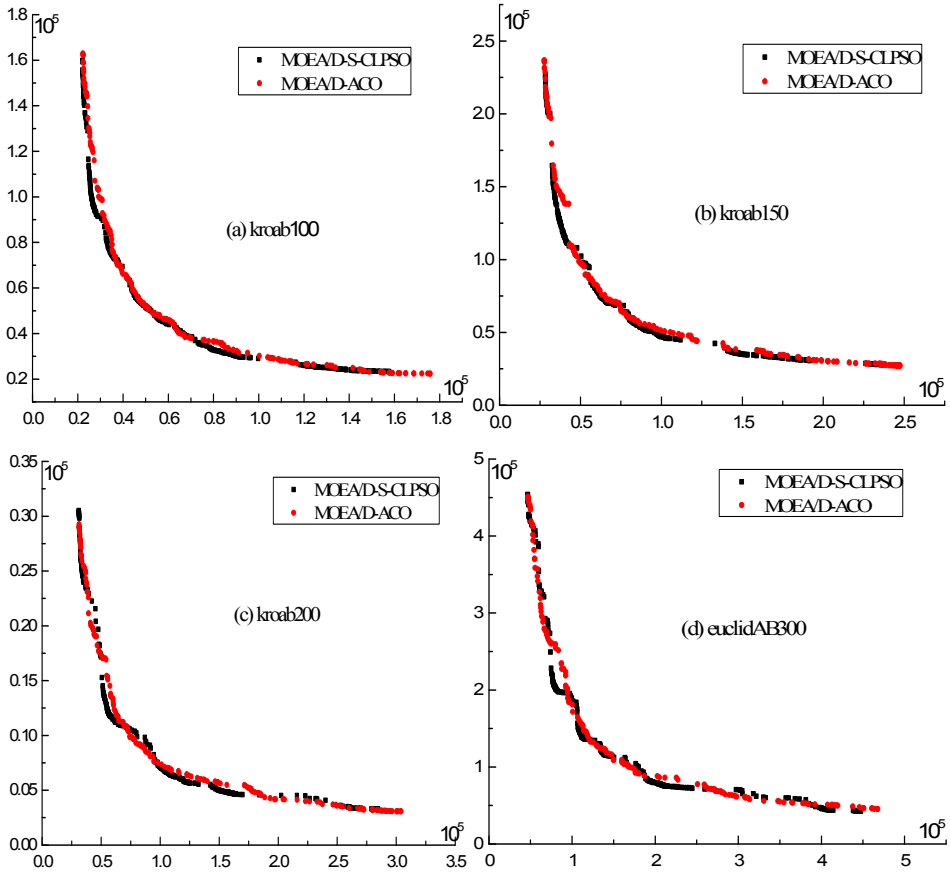



Figure 7. The best approximations with smallest IGD among 30 runs obtained by MOEA-S-CLPSO and MOEA/D-ACO on four TSP instances

Table 1. The IGD statistics obtained by two algorithms over 30 independent runs

| Instances | | MOEA/D-S-CLPSO | | | MOEA/D-ACO | |
|---|---|---|---|---|---|---|
| name | *n* | mean | std | W-test | mean | std |
| kroab100 | 100 | 2061.57 | 325.011 | + | 2332.04 | 189.226 |
| kroac100 | 100 | 2216.75 | 270.048 | + | 2839.58 | 231.312 |
| kroad100 | 100 | 1866.03 | 269.045 | + | 2419.98 | 162.166 |
| kroae100 | 100 | 2030.45 | 301.114 | + | 2644.03 | 183.91 |
| krobc100 | 100 | 1847.77 | 216.977 | + | 2853.49 | 256.169 |
| krobd100 | 100 | 2041.95 | 293.867 | + | 2742.4 | 204.823 |
| krobe100 | 100 | 2150.96 | 214.202 | + | 3016.27 | 236.163 |
| kroab150 | 150 | 4382.99 | 658.298 | — | 4307.53 | 269.1 |
| kroab200 | 200 | 6899.61 | 613.307 | + | 7266.96 | 309.189 |
| euclidAB100 | 100 | 2106.46 | 268.593 | + | 3203.1 | 240.606 |
| euclidAB300 | 300 | 13374.8 | 604.431 | + | 13789.6 | 432.348 |

W-test (Wilcoxon sighed-rand test) is made at the 5% significance level to compare the IGD values of MOEA/D-S-CLPSO and MOEA/D-ACO. "+" and "—" denote that the mean IGD value of the corresponding algorithm is significantly smaller (better) or larger (worse) than that of MOEA/D-ACO.

- $F$ value for evenly-distributed-like weight vectors, $F = 5$
- Refreshing gap $ref\_g = 7$
- $pc[]$, $pc[i] = 0.1$ for each particle $i$
- $c = 2.0$, $w$ linearly decrease within [0.4, 0.9]
- $\alpha = 0.001$ for obtaining $\alpha$ -cut of a set

All statistics are based on 30 independent runs. MOEA/D-ACO stops after 3000 generations with 24 ants and MOEA/D-S-CLPSO stops after 1000 generations with 72 particles, thus the same number of candidate solutions for each MOTSP instance are generated in each run.

## 5.3 Experimental results

The mean and standard deviation of IGDs obtained by MOEA/D-ACO and MOEA/D-S-CLPSO are summarized in Table 1. As shown in Table 1, mean IGD value obtained by MOEA/D-S-CLPSO is significantly smaller than that obtained by MOEA/D-ACO for almost all instances except for kroab150ab, i.e., MOEA/D-SCLPSO can approximates the *PF* better for those MOTSP instances.

However, it can't be ignored that the standard deviation values of MOEA/D-S-CLPSO are larger than MOEA/D-ACO's on all instances apart from krobc100 and krobe100, especially for kroab150, kroab200 and euclidAB300. Combining the results or IGDs' mean and standard deviation, we find that compared with MOEA/D-ACO, MOEA/D-S-CLPSO can achieve better approximations to *PF* for almost all instances but with poorer stability, it can't guarantee its solution quality in an arbitrary run.

For further discussion, the final approximations with smallest IGDs for four MOTSP instances, i.e., kroab100, kroab150, kroab200 and euclidAB300, over 30 runs are given in Figure 7. Although the quality differences between the two algorithms are hard to detect visually, it can be found that the solutions generated by MOEA/D-S-CLPSO are commonly denser in comparison with MOEAD-ACO. Meanwhile, it is clear that nearly all the approximations of two algorithms miss some parts of the real *PF*, which is not desired and hoped to be avoided in further study.

# 6. CONCLUSION

In this paper, a hybrid algorithm namely MOEA/D-S-CLPSO is proposed to solve multiobjective traveling salesman problems by aggregating the set-based CLPSO into the MOEA/D framework. In the hybrid algorithm, an MOTSP is decomposed into a set of single-objective subproblems and each particle is responsible for solving a specific subproblem in S-CLPSO way. Experiments in comparison with MOEA/D-ACO are designed to prove the validity of MOEA/D-S-CLPSO. The statistical results show that a better solution quality can be achieved by MOEA/D-S-CLPSO at most time but the quality can't be guaranteed due to poor stability. Also, the deficiency shared by the two algorithms is addressed in this paper.

# 7. ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation of China (NSFC) Projects No.61379061, No.61332002, No.61202130, in part by Natural Science Foundation of Guangdong No. S2013040014949, in part by Specialized Research Fund for the Doctoral Programs No. 20120171120027, No. 20130171120016, and in part by the Guangzhou Pearl River New Star of Science and Technology No. 151700098.

# 8. REFERENCES

[1] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Berlin: Springer, 2003.

[2] R. Baraglia, J. Hidalgo, and R. Perego, "A hybrid heuristic for the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 6, pp. 613 –622, 2001.

[3] F. Gao, A.-M. Zhou, and G.-X. Zhang, "An estimation of distribution algorithm based on decomposition for the multiobjective TSP," in *Inter. Conf. on Natural Comp. (ICNC 2012)*, May 2012, pp. 817-821.

[4] K. C. Tan, H. J. Tang, and S. S. Ge, "On parameter settings of Hopfield networks applied to traveling salesman

problems," *IEEE Trans. on Circuit and Systems I: Regular Papers*, vol. 52, no. 5, pp. 994-1002, 2005.

[5] J. W. Pepper, B. L. Golden, and E. A. Wasil, "Solving the traveling salesman problem with annealing-based heuristics: a computational study," *IEEE Trans on System, Man and Cybernetic*, vol. 32, no. 1, pp. 72-77, 2002.

[6] T. Weise, R. Chiong, J. Lassig, and K. Tang et al., "Benchmarking optimization algorithms: An open source framework for the traveling salesman problem," *IEEE Trans. on Comput. Intelli. Mag.*, vol. 9, no. 3, pp. 40-52, 2014.

[7] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Implementation of an effective hybrid GA for large-scale traveling salesman problems", *IEEE Trans. Sys., Man, and Cyber.*, vol. 37, no. 1, pp. 92-99, 2007.

[8] K. Miettinen, *Nonlinear Multiobjective Optimization.* Boston, MAA: Kluwer, 1999.

[9] L. J. Ke, Q. F. Zhang, and R. Battiti, "Hybridization of Decomposition and Local Search for Multiobjective Optimization," IEEE Trans. On Cyber., vol. 44, no. 10, pp. 1808-1820, 2014.

[10] P. Larrañaga, and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell: Kluwer, 2001.

[11] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem", *IEEE Trans. Sys., Man, and Cyber.*, vol. 42, no. 5, pp. 682-691.

[12] L. J. Ke, Q. F. Zhang, and R. Battiti, "MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and AntColony," *IEEE Trans. on Cybernetics*, vol. 43, no. 6, pp. 1845-1859, 2013.

[13] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 26, no. 1, pp. 29-41, 1996.

[14] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comp.*, vol. 11, no. 6, pp. 712-731, 2007.

[15] C. W. Chou, C. F. Chien, and M. Gen, "A Multiobjective Hybrid Genetic Algorithm for TFT-LCD Module Assembly Scheduling," *IEEE Trans. Auto. Sci. and Engineering*, vol. 11, no. 3, pp. 692-705, 2014.

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE. Conf. Neur. Net.*, Nov. 1995, pp. 1942-1948.

[17] Y. del Valle, G. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, 2008.

[18] M. Clerc, *Discrete Particle Swarm Optimization*, ser. New Optimization Techniques in Engineering. New York: Springer-Verlag, 2004.

[19] W. Pang, K.-P. Wang, C.-G. Zhou, L.-J. Dong, M. Liu, H.-Y. Zhang, and J.-Y. Wang, "Modified particle swarm optimization based on space transformation for solving traveling salesman problem," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, vol. 4, 2004, pp. 2342–2346.

[20] W. N. Chen, J. Zhang, H. S. H. Chung, W. L. Zhong, W. G. Wu, and Y. H. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Trans. Evol. Comp.*, vol. 14, no. 2, pp. 278-300, 2010.

[21] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Bashar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comp.*, vol. 10, no. 3, pp. 281-295, 2006.

[22] S. Iredi, D. Merkle, and M. Middendorf, "Bi-criterion optimization with multi colony ant algorithm," in *Proc. 1st Int. Conf. EMO*, 2001, pp. 359-372.

[23] C. Garcia-Martinez, O. Cordon, and F. Herrera, "A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP," *Eur. J. Oper. Res.*, vol. 180, no. 1, pp. 116–148, Jul. 2007.

[24] I. Alaya, C. Solnon, and K. Ghedira, "Ant colony optimization for multiobjective optimization problems," in *Proc. 19th IEEE Int. Conf. Tools Artif. Intell.*, 2007.

[25] [Online]. Available: http://eden.dei.uc.pt/~paquete/tsp/

[26] [Online]. Available: http://dces.essex.ac.uk/staff/qzhang/