ADVANCES IN INTELLIGENT SYSTEMS AND COMPUTING 227

Michael Emmerich • André Deutz
Oliver Schütze • Thomas Bäck
Emilia Tantar • Alexandru-Adrian Tantar
Pierre del Moral • Pierrick Legrand
Pascal Bouvry • Carlos Coello Coello (Eds.)

EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV

International Conference held at Leiden University, July 10–13, 2013

Springer

Advances in Intelligent Systems and Computing

Volume 227

Series Editor

J. Kacprzyk, Warsaw, Poland

For further volumes:
http://www.springer.com/series/11156

Michael Emmerich · André Deutz · Oliver Schütze
Thomas Bäck · Emilia Tantar
Alexandru-Adrian Tantar · Pierre del Moral
Pierrick Legrand · Pascal Bouvry
Carlos Coello Coello
Editors

EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV

International Conference held at Leiden University, July 10–13, 2013

Springer

Editors
Michael Emmerich
Leiden Institute of Advanced Computer Science
Leiden University,
Leiden, The Netherlands

André Deutz
Leiden Institute of Advanced Computer Science
Leiden University, Leiden, The Netherlands

Oliver Schütze
CINVESTAV-IPN
Depto. De Computación
Mexico City, Mexico

Thomas Bäck
Leiden Institute of Advanced Computer Science
Leiden University, Leiden, The Netherlands

Emilia Tantar
Luxembourg Centre for Systems Biomedicine
University of Luxembourg
Belval, Luxembourg

Alexandru-Adrian Tantar
Computer Science and Communications
Research Unit
University of Luxembourg
Luxembourg

Pierre del Moral
Bordeaux Mathematical Institute
Université Bordeaux I
Talence cedex, France

Pierrick Legrand
Université Bordeaux Segalen,
UFR Sciences et Modélisation
Bordeaux, France

Pascal Bouvry
Faculty of Sciences, Technology and
Communication
Computer Science and Communication Group
University of Luxembourg
Luxembourg

Carlos Coello Coello
CINVESTAV-IPN
Depto. de Computatción
Mexico City, Mexico

ISSN 2194-5357 ISSN 2194-5365 (electronic)
ISBN 978-3-319-01127-1 ISBN 978-3-319-01128-8 (eBook)
DOI 10.1007/978-3-319-01128-8
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012944264

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The overarching goal of the EVOLVE international conference series is to build a bridge between probability, statistics, set oriented numerics and evolutionary computing, as to identify new common and challenging research aspects and solve questions at the cross-sections of these fields. There is a growing interest for large-scale computational methods with robustness and efficiency guarantees. This includes the challenge to develop sound and reliable methods, a unified terminology, as well as theoretical foundations.

In this year's edition of EVOLVE held at LIACS, Leiden University, The Netherlands (evolve2013.liacs.nl) major themes are machine learning, probabilistic systems, complex networks, genetic programming, robust and diversity-oriented and multiobjective optimization using evolutionary methods as well as set-oriented numerics and cell-mapping. The range of topics is also well reflected in the spectrum of special sessions, organized by internationally renowned experts: *Set Oriented Numerics* (Michael Dellnitz and Kathrin Padberg), *Evolutionary Multiobjective Optimization* (Oliver Schütze), *Genetic Programming* (Pierrick Legrand, Leonardo Trujillo and Edgar Galvan), *Probabilistic Models and Algorithms: Theory and Applications* (Arturo Hernández-Aguirre), *Diversity-oriented Optimization* (Vitor Basto Fernandes, Ofer Shir, Iryna Yevseyeva, Michael Emmerich, and André Deutz), *Complex Networks and Evolutionary Computation* (Jing Liu), *Robust Optimization* (Massimiliano Vasile), and *Computational Game Theory* (Rodica I. Lung and D. Dumitrescu).

The twenty peer reviewed contributions discuss overarching themes and present answers to questions, such as: How to assess performance? How to design reliable and efficient computational algorithms for learning, genetic programming, and multiobjective/multimodal/robust optimization? What are the sources of complexity and problem difficulty? How to build a frameworks for set-oriented

search? Therefore these proceedings present cutting edge research in these related fields and will serve as a stepping stone towards a more integrated view of advanced computational models and methods.

Leiden, Mexico City, Luxembourg, and Bordeaux
April 2013

Michael Emmerich
André Deutz
Oliver Schütze
Thomas Bäck
Emilia Tantar
Alexandru-Adrian Tantar
Pierre del Moral
Pierrick Legrand
Pascal Bouvry
Carlos Coello Coello

Organization

EVOLVE 2013 is organized by the *Leiden Institute for Advanced Computer Science (LIACS), Faculty of Science*, Leiden University, the Netherlands in cooperation with INRIA, France, CINVESTAV, Mexico, University of Bordeaux, INRIA Bordeaux Sud-Ouest, and the University of Luxembourg, Computer Science and Communication Research Unit.

Executive Committee

General Chair

Michael Emmerich	Leiden University, nl

Local Chairs

André Deutz	Leiden University, nl
Thomas Bäck	Leiden University, nl

Series Chairs

Oliver Schütze	CINVESTAV-IPN, Mexico City, mx
Emilia Tantar	University of Luxembourg, lu
Alexandru Tantar	University of Luxembourg, lu
Pierrick Legrand	University of Bordeaux 2, fr
Pierre del Moral	INRIA Bordeaux Sud-Ouest, fr
Pascal Bouvry	University of Luxembourg, lu

Local Support

Marloes van der Nat	Leiden University, nl
Irene Nooren	Leiden University, nl
Zhiwei Yang	Leiden University, nl
Edgar Reehuis	Leiden University, nl

Publicity Chair

Rui Li	Leiden University, nl

Art Design

Adriana Martínez	KREAPROM, Mexico City, mx

Advisory Board

Enrique Alba	University of Málaga, es
François Caron	INRIA Bordeaux Sud-Ouest, fr
Frédéric Ciérou	INRIA Rennes Bretagne Atlantique, fr
Carlos A. Coello Coello	CINVESTAV-IPN, mx
Michael Dellnitz	University of Paderborn, de
Frédéric Guinand	University of Le Havre, fr
Arnaud Guyader	Université Rennes 2, INRIA Rennes Bretagne Atlantique, fr
Arturo Hernández-Aguirre	CIMAT, Guanajuato, mx
Günter Rudolph	TU Dortmund, de
Marc Schoenauer	INRIA Saclay – Île-de-France, Université Paris Sud, fr
Franciszek Seredynski	Polish Academy of Sciences, Warsaw, pl
El-Ghazali Talbi	Polytech'Lille University of Lille 1, fr
Marco Tomassini	University of Lausanne, ch
Massimiliano Vasile	University of Strathclyde, uk

Special Session Chairs

Oliver Schütze	CINVESTAV, Mexico City, mx
Michael Dellnitz	University of Paderborn, de
Kathrin Padberg	TU Dresden, de
Pierrick Legrand	University of Bordeaux 2, fr
Leonardo Trujillo	ITT, mx
Edgar Galvan	Trinity College, ie
Arturo Hernández-Aguirre	CIMAT, Guanajuato, mx
Vitor Basto Fernandes	CIIC, pt
Ofer Shir	IBM Research, il
Iryna Yevseyeva	Newcastle University, uk
Michael Emmerich	Leiden University, nl
André Deutz	Leiden University, nl
Jing Liu	Xidian University, cn
Rodica Ioana Lung	Babes-Bolyai University, ro
Dumitru Dumitrescu	Babes-Bolyai University, ro
Massimiliano Vasile	University Strathclyde, uk

Program Committee

Conference Chair

Michael Emmerich	Leiden University, nl

Program Chairs

Michael Emmerich	Leiden University, nl
André Deutz	Leiden University, nl
Oliver Schütze	CINVESTAV, mx
Thomas Bäck	Leiden University, nl
Emilia Tantar	University of Luxembourg, lu
Alexandru Tantar	University of Luxembourg, lu
Pierrick Legrand	University of Bordeaux 2, fr
Pierre del Moral	INRIA Bordeaux Sud-Ouest, fr
Pascal Bouvry	University of Luxembourg, lu

All full papers were thoroughly peer reviewed. We thank the referees for their voluntary effort.

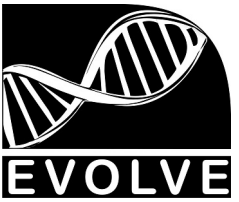
Referees

Josiah Adeyemo
Gideon Avigad
Vitor Manuel Basto Fernandes
Urvesh Bhowan
Nohe R. Cazaez-Castro
Francisco Chicano
Brendan Cody-Kenny
Nareli Cruz
Liliana Cucu-Grosjean
Luis Gerardo De La Fraga
Michael Dellnitz
André Deutz
Jianguo Ding
Christian Domínguez Medina
Dumitru Dumitrescu
Enrique Dunn
Erella Eisenstadt
Michael Emmerich
Francisco Fernandez
Edgar Galvan
Arturo Hernández Aguirre
Ahmed Kattan
Timoleon Kipouros
Joanna Kolodziej
Angel Kuri-Morales
Adriana Lara
Pierrick Legrand
Rui Li
Jing Liu
Francisco Luna
Rodica Ioana Lung
Gabriel Luque
Nashat Mansour
Jörn Mehnen
James McDermott
Nicolas Monmarché
Sanaz Mostaghim
Boris Naujoks
Sergio Nesmachnow
Gustavo Olague
Kathrin Padberg-Gehle
Eunice E. Ponce-de-Leon
Marcus Randall
Edgar Reehuis
Eduardo Rodriguez-Tello
Günter Rudolph
Thomas Schaberreiter
Christoph Schommer
Ofer Shir
Ignacio Segovia Dominguez
Oliver Schütze
Sara Silva
Leonardo Trujillo
Cliodhna Tuite
Massimiliano Vasile
Sergio Ivvan Valdez
Fatos Xhafa
Iryna Yevseyeva

Sponsoring Institutions

LIACS, Faculty of Science, Leiden University, nl
CINVESTAV-IPN, mx
NWO, nl
University of Luxembourg, lu
University of Bordeaux, fr
INRIA Bordeaux Sud-Ouest, fr

Conference Logo



Contents

Machine Learning and Probabilistic Models

Constructing a Solution Attractor for the Probabilistic Traveling Salesman Problem through Simulation 1
Weiqi Li

Unsupervised Classifier Based on Heuristic Optimization and Maximum Entropy Principle 17
Edwin Aldana-Bobadilla, Angel Kuri-Morales

Training Multilayer Perceptron by Conformal Geometric Evolutionary Algorithm 31
J.P. Serrano, Arturo Hernández, Rafael Herrera

RotaSVM: A New Ensemble Classifier 47
Shib Sankar Bhowmick, Indrajit Saha, Luis Rato, Debotosh Bhattacharjee

Complex Networks and Evolutionary Computation

Finding Biologically Plausible Complex Network Topologies with a New Evolutionary Approach for Network Generation 59
Gordon Govan, Jakub Chlanda, David Corne, Alex Xenos, Pierluigi Frisco

Fitness Landscape Analysis of NK Landscapes and Vehicle Routing Problems by Expanded Barrier Trees 75
Bas van Stein, Michael Emmerich, Zhiwei Yang

Sewer Network Design Optimization Problem Using Ant Colony Optimization Algorithm and Tree Growing Algorithm 91
Ramtin Moeini, Mohammed Hadi Afshar

An Ant-Based Optimization Approach for Inventory Routing ... 107
Vasileios A. Tatsis, Konstantinos E. Parsopoulos, Konstantina Skouri, Ioannis Konstantaras

Diversity Oriented Optimization

Measuring Multimodal Optimization Solution Sets with a View to Multiobjective Techniques ... 123
Mike Preuss, Simon Wessing

A Benchmark on the Interaction of Basic Variation Operators in Multi-objective Peptide Design Evaluated by a Three Dimensional Diversity Metric and a Minimized Hypervolume ... 139
Susanne Rosenthal, Markus Borschbach

Logarithmic-Time Updates in SMS-EMOA and Hypervolume-Based Archiving ... 155
Iris Hupkens, Michael Emmerich

Set-Oriented Numerics and Evolutionary Multiobjective Optimization

Computing the Set of Approximate Solutions of a Multi-objective Optimization Problem by Means of Cell Mapping Techniques ... 171
Carlos Hernández, Jian-Qiao Sun, Oliver Schütze

The Directed Search Method for Pareto Front Approximations with Maximum Dominated Hypervolume ... 189
Víctor Adrián Sosa Hernández, Oliver Schütze, Günter Rudolph, Heike Trautmann

A Hybrid Algorithm for the Simple Cell Mapping Method in Multi-objective Optimization ... 207
Yousef Naranjani, Carlos Hernández, Fu-Rui Xiong, Oliver Schütze, Jian-Qiao Sun

Planning and Allocation Tasks in a Multicomputer System as a Multi-objective Problem ... 225
Apolinar Velarde, Eunice Ponce de León, Elva Diaz, Alejandro Padilla

Real World System Architecture Design Using Multi-criteria Optimization: A Case Study ... 245
Rajesh Kudikala, Andrew R. Mills, Peter J. Fleming, Graham F. Tanner, Jonathan E. Holt

A Self-adaptive Genetic Algorithm Applied to Multi-Objective Optimization of an Airfoil ... 261
John M. Oliver, Timoleon Kipouros, A. Mark Savill

Genetic Programming

Genetic Programming with Scale-Free Dynamics 277
Hitoshi Araseki

Preliminary Study of Bloat in Genetic Programming with Behavior-Based Search 293
Leonardo Trujillo, Enrique Naredo, Yuliana Martínez

Robust Optimization

Evidence Theory Based Multidisciplinary Robust Optimization for Micro Mars Entry Probe Design 307
Liqiang Hou, Yuanli Cai, Rongzhi Zhang, Jisheng Li

Author Index 323

Constructing a Solution Attractor for the Probabilistic Traveling Salesman Problem through Simulation

Weiqi Li

University of Michigan – Flint 303 E. Kearsley Street,
Flint, MI Postal Code 48502, U.S.A.
weli@umflint.edu

Abstract. The probabilistic traveling salesman problem (PTSP) is a variation of the classic traveling salesman problem and one of the most significant stochastic network and routing problems. This paper proposes a simulation-based multi-start search algorithm to construct the solution attractor for the PTSP and find the best *a priori* tour through the solution attractor. A solution attractor drives the local search trajectories to converge into a small region in the solution space, which contains the most promising solutions to the problem. Our algorithm uses a simple multi-start local search process to find a set of locally optimal *a priori* tours, stores these tours in a so-called hit-frequency matrix E, and then finds a globally optimal *a priori* tour in the matrix E. In this paper, the search algorithm is implemented in a master-worker parallel architecture.

Keywords: Stochastic optimization, global optimization, simulation, probabilistic traveling salesman problem, local search, parallel algorithm.

1 Introduction

The classic traveling salesman problem (TSP) is defined as: Given a set of n cities and an $n \times n$ cost matrix C in which $c(i, j)$ denotes the traveling cost between cities i and j ($i, j = 1,2,\ldots, n$; $i \neq j$). A tour π is a closed route that visits every city exactly once and returns at the end to the starting city. The goal is to find a tour π^* with minimal traveling cost.

In the real world, many optimization problems are inherently dynamic and stochastic. Such problems exist in many areas such as optimal control, logistic management, scheduling, dynamic simulation, telecommunications networks, genetics research, neuroscience, and ubiquitous computing. As real-time data in information systems become increasingly available with affordable cost, people have to deal with more and more such complex application problems. For the TSP under dynamic and stochastic environment, the number of cities n can increase or decrease and the cost $c(i, j)$ between two cities i and j can change with time. In this paper we consider only the case in which the number of cities n changes with time t. Therefore, the TSP can be defined as:

M. Emmerich et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV*, Advances in Intelligent Systems and Computing 227,
DOI: 10.1007/978-3-319-01128-8_1,

$$\min \quad f(\pi) = \sum_{i=1}^{n_t - 1} c(i,\ i+1) + c(n_t,\ 1) \tag{1}$$

$$\text{subject to} \quad n_t \in N$$

where n_t is the number of cities at time t and N is the set of all potential cities existing in the problem. If we want to design an algorithm and its purpose is to continuously track and adapt the changing n through time and to find the currently best solution quickly, that is, to re-optimize solution for every change of n, the TSP is defined as a dynamic TSP. If we treat the number of cities n as a random variable and wish to find an *a priori* tour through all N cities, which is of minimum cost in the expected value sense, the TSP becomes a probabilistic TSP (PTSP). In a PTSP, on any given realization of the problem, the n cities present will be visited in the same order as they appear in the *a priori* tour, i.e., we simply skip those cities not requiring a visit. The goal of an algorithm for PTSP is to find a feasible *a priori* tour with minimal expected cost [1, 2].

The PTSP was introduced by Jaillet [2, 3], who examined some of its combinatorial properties and derived a number of asymptotic results. Further theoretical properties, asymptotic analysis and heuristic schemes have been investigated by [4-7]. Surveys of approximation schemes, asymptotic analysis, and complexity theorems for a class of a priori combinatorial optimization problems can be found in [1, 8].

Formally, a PTSP is defined on a complete graph $G = (V,\ A,\ C,\ P)$, where $V = \{v_i\colon i = 1, 2, \ldots, N\}$ is a set of nodes; $A = \{a(i, j)\colon i, j \in V, i \neq j\}$ is the set of edges that completely connects the nodes; $C = \{c(i, j)\colon i, j \in A\}$ is a set of costs associated with the edges; $P = \{p_i\colon i \in V\}$ is a set of probabilities that for each node v_i specifies its probability p_i of requiring a visit. In this paper, we assume that the costs are symmetric, that is, traveling from a node v_i to v_j has the same cost as traveling from node v_j to v_i. We also assign v_1 as the depot node with the presence probability of 1.0. Each non-depot node v_i is associated with a presence probability p_i that represents the possibility that node v_i will be present in a given realization. Based on the values of presence probability (p_i) of non-depot nodes, two types of PTSP can be classified: the homogeneous and heterogeneous PTSP. In the homogeneous PTSP, the presence probabilities of non-depot nodes are all equal ($p_i = p$ for every non-depot node v_i); in the heterogeneous PTSP, these probabilities are not necessarily the same.

Designing effective and efficient algorithms for solving PTSP is a really challenging task, since in PTSP, the computational complexity associated with the combinatorial explosion of potential solutions is exacerbated by the stochastic element in the data. The predominant approach to finding good solutions for PTSP instances has been the adaptation of TSP heuristics [5-8]. In general, researchers use two techniques in their search algorithms: analytical computation and empirical estimation [9]. The analytical computation approach computes the cost $f(\pi)$ of an *a priori* tour π, using a closed-form expression. Empirical estimation simply estimates the cost through placeMonte Carlo simulation. Birattari

et al. [9] discussed some limitations on analytical computation technique and suggested that the empirical estimation approach can overcome the difficulties posed by analytical computation.

In recent years, many local search and metaheuristic algorithms such as ant colony optimization, evolutionary computation, simulated annealing and scatter search, using analytical computation or empirical estimation approach, have been proposed to solve the PTSP [4, 7, 10-21]. Bianchi et al. [22] provided a comprehensive overview about recent developments in the metaheuristic algorithms field. In this paper, we propose a new simulation-based algorithm to solve PTSP. This paper is organized as follows. In section 2 we discuss the placeMonte Carlo sampling approximation. In section 3, we briefly describe the construction of solution attractor for TSP. In section 4, we describe the proposed simulation-based algorithm and discuss some experimental results. Section 5 concludes this paper.

2 Monte Carlo Sampling Approximation

The sampling approximation method is an approach for solving stochastic optimization problem by using simulation. The given stochastic optimization problem is transformed into a so-called sample average optimization problem, which is obtained by considering several realizations of the random variable and by approximating the cost of a solution with a sample average function [23]. In the context of the PTSP, this method has been shown to be very effective [9, 24].

In sampling approximation search, a stochastic optimization problem is represented by a computer simulation model. Simulation models are models of real or hypothetical systems, reflecting all important characteristics of the system under studied. Perhaps one of the best known methods for sampling a probability distribution is the placeMonte Carlo sampling technique, which is based on the use of a pseudo random number generator to approximate a uniform distribution. Currently, the placeMonte Carlo sampling approximation method is the most popular approach for solving stochastic optimization problems. In this technique the expected objective function of the stochastic problem is approximated by a sample average estimate derived from a random sample. We assume that the sample used at any given iteration is independent and identically distributed, and that this sample is independent of previous samples. The resulting sample average approximating problem is then solved by deterministic optimization techniques. The process can be repeated with different samples to obtain candidate solutions along with statistical estimates of their optimality gaps [18]. Sampling approximation via simulation is statistically valid in the context of simulation as the underlying assumptions of normality and independence of observations can be easily achieved through appropriate sample averages of independent realizations, and through adequate assignment of the pseudo-random number generator seeds, respectively.

In the case of PTSP, the elements of the general definition of the stochastic problem take the following format: a feasible tour π is an *a priori* tour visiting once and only once all N cities, and the random variable n is extracted from an N-variate Bernoulli distribution and prescribes which cities need being visited. This leads in total of 2^N possible scenarios for N cities. A nave way to calculate the expected cost of a solution would be to sum over all possible scenarios the cost of the *a posteriori* solution in this scenario multiplied with the probability for this scenario. Obviously the summation over 2^N terms is computationally intractable for reasonable values of N. Another more efficient way is using placeMonte Carlo sampling: instead of summing over all possible scenarios, we could sample M ($M < 2^N$) scenarios of using the known probabilities and take the average over the costs of the *a posteriori* tours for the sampled scenarios. Therefore, the cost $f(\pi)$ of a PTSP tour π can be empirically estimated on the basis of a sample $f(\pi, n_1)$, $f(\pi, n_2)$, ..., $f(\pi, n_M)$ of costs of *a posteriori* tours obtained from M independent realizations n_1, n_2, ..., n_M of the random variable n:

$$\hat{f}_M(\pi) = \tfrac{1}{M} \sum_{i=1}^{M} f(\pi, n_i) \tag{2}$$

$\hat{f}_M(\pi)$ denotes the average of the objective values of the M realizations on the *a priori* tour π, which gives us an approximation for the estimated cost for tour π. Clearly, $\hat{f}_M(\pi)$ is an unbiased estimator of $f(\pi)$. A search algorithm for PTSP is looking for the optimal tour π^* which has the smallest estimated objective value$\hat{f}_M(\pi^*)$, that is,

$$\pi^* \in \arg\min\{\hat{f}_M(\pi_1), \hat{f}_M(\pi_2), \ldots\} \tag{3}$$

The optimal value $\hat{f}_M(\pi^*)$ and the optimal tour π^* to the PTSP provide estimates of their true counterparts.

Because we obtain only estimates using this way, it may not be possible to decide with certainty whether tour π_i is better than tour π_j, which may frustrates the search algorithm that tries to move in an improving direction. In principle, we can eliminate this complication by making so many replications at each iterative point that the performance estimate has essentially no variance. In practice, this could mean that we will explore very few iteration due to the time required to simulate each one. Therefore, in a practical sampling approximation algorithm, the test if a solution is better than another one can only be done by statistical sampling, that is, obtaining a correct comparison result only with a certain probability. The goal now is to get a good average case solution and the expected value of the objective is to be optimized. The way simulation approximation is used in an optimization algorithm largely depends on the way solutions are compared and the best solutions among a set of other solutions is selected.

The number of realizations M should be large enough for providing a reliable estimate of the cost of solutions but at the same time it should not be too large otherwise too much time is wasted. The appropriate number of realizations M depends on the stochastic character of the problem at hand. The larger the

probability that a city is to be visited, the less stochastic an instance is. In this case, the algorithms that obtain the best results are those that consider a reduced number of realizations and therefore explore more solutions in the unit of time. On the other hand, when the probability that a city is to be visited is small, the instance at hand is highly stochastic. In this case, it pays off to reduce the total number of solutions explored and to consider a larger number of realizations for obtaining more accurate estimates [23, 25, 26].

There are many sampling strategies available. For the PTSP, common sampling strategies include (1) the same set of M realizations is used for all steps of the iteration in the algorithm; (2) a set of M realizations is sampled anew each time an improved solution is found; and (3) a set of M realizations is sampled anew for each comparison of solutions. The first strategy is a well-known variance-reduction technique called the method of common random numbers (CRN). CRN takes advantage of the same set of random numbers across all alternatives for a given replication. CRN is typically designed to induce positive correlation among the outputs of each respective alternative for a given replication, thereby reducing the variance of the difference between the mean alternative point estimators. One of the practical motivations for using CRN in a search algorithm is to speed up the sample average computations. However, one major problem with CRN is that the iterates of the algorithm may be "trapped" in a single "bad" sample path. Second and third strategies are called variable-sample method. Resampling allow the iterates of the algorithm to get away from those "bad" sample paths. Another advantage of a variable-sample scheme is that the sample sizes can increase along the algorithm, so that sampling effort is not wasted at the initial iteration of the algorithm [27].

Several researchers have been proposed estimation-based algorithms to deal with the PTSP, using local search or metaheuristics [21, 24-28]. This paper introduces a new optimization approach by using solution-attractor construction in the context of placeMonte Carlo simulation. Our algorithm includes optimization and simulation in a parallel iterative process in order to gain the advantages of optimization (exact solution), simulation (stochasticity) and speed (parallel processing).

3 Solution Attractor Construction

Our approach uses a parallel multi-start search procedure to construct the solution attractor for the PTSP. Due to the NP-hardness and intractability of the combinatorial optimization problems, heuristic search techniques have become a popular means to find reasonable good solutions to these problems. Many search heuristics have been based on or derived from a general technique known as local search [29]. A *search trajectory* is the path followed by the search process in the solution space as it evolves with time. Local search techniques are locally convergent. The final solution usually is a locally optimal solution. Local optimality

depends on the initial solution and the neighborhood function that is used in the search process. In order to overcome local optimality, heuristic search usually require some type of diversification to avoid a large region of the solution space remaining completely unexplored. One simple way to achieve this diversification is to start the search process from several different initial points. Multi-start heuristics produce several local optima, and the best overall is the algorithm's output. Multi-start search helps to explore different areas in the solution space and therefore it generates a wide sample of the local optima.

For some optimization problem such as TSP, these multi-start search trajectories will converge to a small region in the solution space. From dynamic system perspective, this small region is called a *solution attractor* [30]. The solution attractor of a heuristic search algorithm on an optimization problem is defined as a subset of the solution space that contains the whole solution space of the end points (local optima) of all local search trajectories. Fig. 1 illustrates the concepts of local search trajectories and solution attractor. Since the globally optimal point is a special case of locally optimal points, it is expected to be embodied in the solution attractor. Li [30-32] developed a multi-start search approach to construct the solution attractor for a TSP and applied this approach to tackle multi-objective TSP and dynamic TSP. This paper applies this attractor-construction technique to solve PTSP.

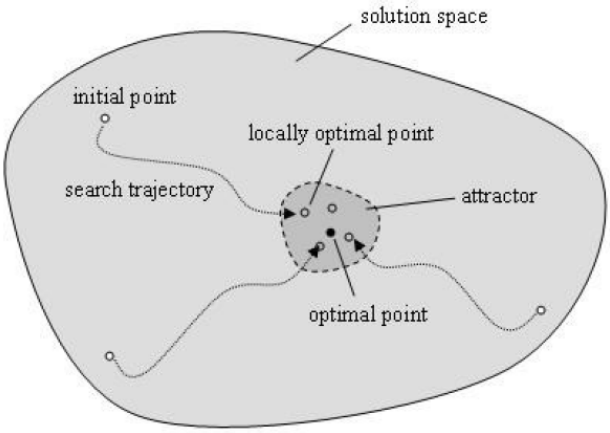


Fig. 1. The concepts of local search trajectories and solution attractor

For a TSP instance, the solution space contains all tours that a salesman may traverse. If we start several distinct search trajectories from different initial points, after letting the search process run for a long time, these trajectories would settle into the same attractive region (the solution attractor) if the problem has only one optimal solution. Fig. 2(a) presents a sequential procedure for constructing the solution attractor of local search in a TSP instance and

Fig. 2(b) shows a parallel procedure implemented in a master-worker architecture. The construction procedure is very straightforward: generating K locally optimal tours, storing them into a matrix (called *hit-frequency matrix* E), removing some unfavorable edges in E if necessary, and then finding all tours contained in E.

The hit-frequency matrix E plays a critical role in the construction of solution attractor. When each search trajectory reaches its locally optimal point, it leaves its "final footprint" in E. That is, E is used to record the number of hits on the edge $a(i, j)$ of the graph by the set of K locally optimal tours. Therefore, E can provide the architecture that allows individual tours to be linked together along a common structure and generate the structure for the global solution. This structure can help us to find the globally optimal tour.

```
1  procedure TSP_Attractor(C)
2  begin
3    repeat
4      π_i = Initial_Tour() ;
5      π_j = Local_Search(π_i) ;
6      Update(E(π_j)) ;
7    until number_of_local_optima = Q ;
8    E = Remove_Noise(E) ;
9    Exhausted_Search(E) ;
10 end
```

(a)

```
1 procedure TSP_Attractor_Workers(C)
2 begin
3   for worker processor 1, …, K
4       π_i = Initial_Tour() ;
5       π_j = Local_Search(π_i) ;
6       sends π_j to master processor ;
7 end


1 procedure TSP_Attractor_Master(E)
2 begin
3   initialize E ;
4   repeat
5     receive a tour from a worker processor
6     update E ;
7   until number_of_local_optima = Q
8 Exhausted_Search(E) ;
9 end
```

(b)

Fig. 2. The procedure for constructing solution attractor of local search in TSP

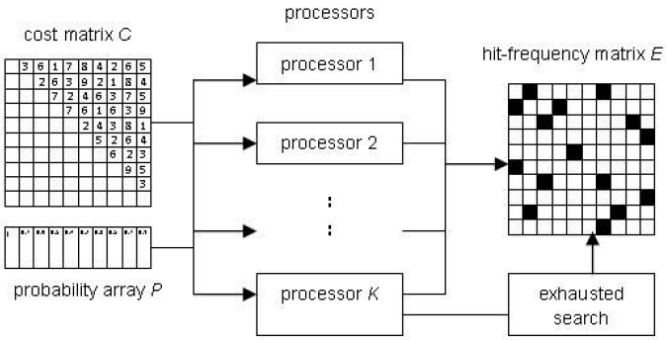


Fig. 3. Schematic structure of the search system for the PTSP

4 The Proposed Simulation-Based Search System

4.1 The Parallel Search System

We implemented our multi-start search system into a parallel search system. Parallel processing can be useful to efficiently solve difficult optimization problems, not only by seeding up the execution times, but also improving the quality of the final solutions. Fig. 3 sketches the basic idea of our parallel search system for the PTSP. This search system contains K processors and bears intrinsic parallelism in its features. Based on a common cost matrix C and probability array P, this system starts K separate search trajectories in parallel. When a search trajectory reaches its locally optimal solution, the processor stores the solution in the common hit-frequency matrix E. The processor starts a new search if more computing time is available. Finally, at the end of the search, the matrix E is searched by an exhausted search process and the best solution in the attractor is outputted as the optimal solution.

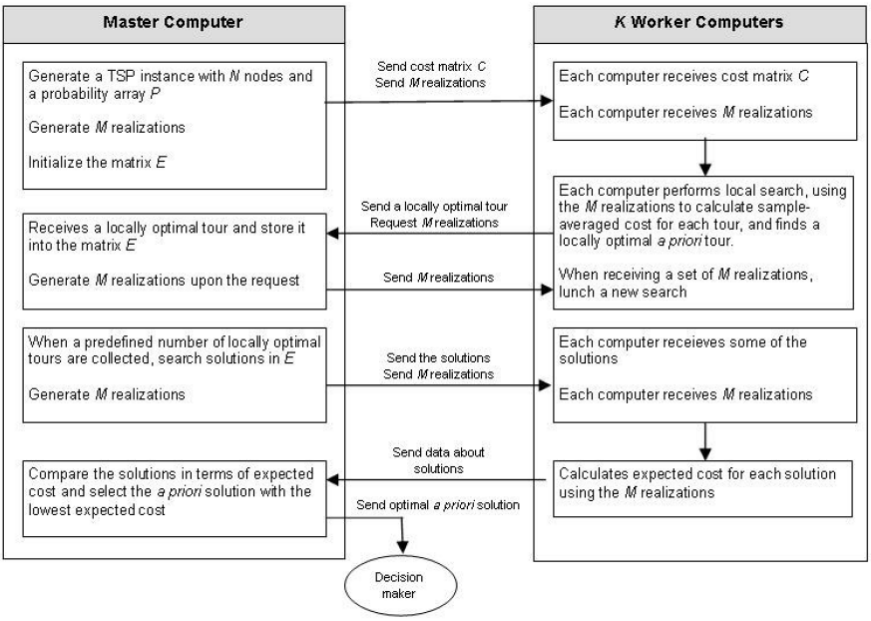


Fig. 4. The search system implemented in a master-worker architecture

Fig. 4 shows the implementation of our search system using a master-worker architecture. In our master-worker architecture, one computer serves as a master and other computers as workers. The master computer generates a TSP instance with N nodes and a probability array P, initializes the hit-frequency matrix E, and sends a copy of the cost matrix C to each of the worker computers.

The multi-search task is distributed to K worker computers. Then based on the probability values in the array P, the master computer generate a set of M realizations (m_1, m_2, ..., m_M). Each of the M realizations is an array that contains binary values, where a value "1" at position i in the array indicates that node v_i requires being visited whereas a value "0" means that it does not require being visited. The master computer sends the set of M realizations to the worker computers. Depending on the setting, the master computer can send the same set of M realizations to all worker processors or a different set of M realizations to each of the worker computers. When receiving the set of M realizations, each worker computer independently performs its local search: it randomly generates an initial *a priori* tour, calculates the sample-averaged cost by using the M realizations, and then generates a new *a priori* tour, calculates its sample-averaged cost by using the same M realizations; if this new a priori tour has lower average cost, the current *a priori* tour is replaced by this better *a priori* tour; otherwise another new *a priori* tour is generated and compared with the current *a priori* tour. When a worker computer reaches a locally optimal *a priori* tour, it sends the tour to the master processor and requests a new set of M realizations from the master processor. When the master processor receives a locally optimal *a priori* tour from a worker processor, it stores the tour into the matrix E. It then generates a new set of M realizations and sends it to the requesting worker computer. In such was, the multi-start search is performed by the K worker computers in parallel. When the worker computer receives the new set of M realizations, it lunches new local search. When a predefined number of locally optimal *a priori* tours are collected from the worker computers and stored in the matrix E, these locally optimal tours form a solution attractor for the problem. The master computer sends a signal to worker computers to stop their searching. It then launches an exhausted search process in the matrix E. After finding all tours in E, the master computer sends these tours to the worker computers with a common set of M realizations; each worker computer receives different tours. The worker computers use the set of M realizations to calculate the sampling average and standard deviation for each of the tours, and send these values back to the master computer. The master computer compares these tours in terms of sampling average and finally outputs the best *a priori* tour.

4.2 The Experimental Problem

Due to the novelty of the field of PTSP there is no test problem instance available to be used for implementing a new search algorithm and assessing its suitability, effectiveness and efficiency. We design a test problem for our experiments.

The design of a test problem is always important in designing any new search algorithm. The context of problem difficulty naturally depends on the nature of problems that the underlying algorithm is trying to solve. In the context of solving PTSP, our test problem was designed based on several considerations. First, the size of problem should be large, since TSP instances as small as 200 cities are now considered to be well within the state of the global optimization art. The instance must be considerable large than this for us to be sure that

the heuristic approach is really called for. Second, there is no any pre-known information related to the result of the experiment, since in a real-world problem one does not usually have any knowledge of the solutions. Third, when dealing with stochastic combinatorial problems, randomness in both process and data means that the underlying model in the algorithm is suitable for the modeling of natural problems. We should formulate a problem that goes nearer to real world conditions. Last, the problem instance should be general, understandable and easy to formulate so that the experiments are repeatable and verifiable.

We set up a TSP instance with $N = 1000$ cities. In the cost matrix C, $c(i, j)$ is assigned a random integer number in the range [1, 1000], where $c(i, j) = c(j, i)$. A probability array P contains a set of probability values that assigns probability p_i to city i. We specify city 1 as the depot node with $p_1 = 1.0$. The probability of each non-depot city is generated from a uniform random number in a certain range. Therefore, our test problem is a heterogeneous PTSP.

4.3 The Experiments

Our experiments were conducted on a network of 6 PCs: Pentium 4 at 2.4 GHz and 512 MB of RAM running Linux, interconnected with a Fast Ethernet communication network using the LAM implementation of the MPI standard. The search system was developed using Sun Java JDK 1.3.1. The network was not dedicated, but was very steady. In our parallel search system, one computer serves as a master computer and other five computers as worker processors. Our network architecture and algorithms are asynchronous by nature, meaning that the processors do not have a common clock by which to synchronize their processing and calculation.

Our experiments relied heavily on randomization. All initial tours were randomly constructed. We used simple 2-opt local search in our search system. During the local search, the search process randomly selected a solution in the neighborhood of the current solution. A move that gave the first improvement was chosen. The local search process in each search trajectory terminated when no improvement could be achieved after 1000 iterations.

Because our search system uses simulation, it is not possible to decide with certainty whether a solution is better than another during the search. This can only be tested by statistical sampling, obtaining a correct comparison result only with a certain probability. In other words, the simulation estimates should be accompanied with some indication of precision. The first decision we had to make in our experiment was to choose an appropriate sample size M (number of realizations) in our simulation. The accuracy factors we considered include desired precision of results, confidence level and degree of variability. We used the following equation to determine our sample size [33, 34]:

$$M = \frac{V(1-V)}{\frac{A^2}{Z^2} + \frac{V(1-V)}{P}} \tag{4}$$

where M is the required sample size; P is the size of population; V is the estimated variance in population, which determines the degree to which the attributes being measured in the problem are distributed throughout the population; A is the desired precision of results that measures the difference between the value of the sample and the true value of the real population, called the sampling error; Z is the confidence level that measures the percentage of the samples would have the true population value within the range of chosen precision; and P is the size of population.

In our search system, we have two phases of simulation. In the first phase, the worker computers use simulation to calculate sample averages and compare tours. In this phase, we choose the sampling error $A = \pm 3\%$ and confidence level $Z = 1.96$ (95% confidence). Because our PTSP instance is heterogeneous, we use variability $V = 50\%$. Using Eq. (4) we calculate $M = 1067$. In the second phase, the workers use simulation to calculate sample averages for the tours found in the matrix E, and then the master computer uses these information to order the tours. In this phase, we choose the sample error $A = \pm 3\%$, confidence level $Z = 2.57$ (99% confidence) and $V = 50\%$. We calculate $M = 1835$. Therefore, in our experiment, we used $M_I = 1100$ realizations in the first phase and $M_{II} = 1850$ in the second phase.

We generated a TSP instance and a probability array P, in which the value of p_i was generated from a uniform random number in the range [0.1, 0.9]. When the master computer collected 30 locally optimal tours from the worker computers, the search system stopped searching. The master computer applied an exhausted-search procedure on E, and found 36 tours in E. The master computer sent these tours to the worker computers. Four worker computers got 7 tours and one worker computer got 8 tours. The worker computers calculate the sampling averages and standard deviations for these tours. Table 1 lists the five best tours found in E. We can see that the expected cost of the best tour is 4889 with standard deviation 201.

Table 1. The five best torus found in the matrix E

Tour	Sampling Average	Standard Deviation
1	4889	201
2	4902	224
3	5092	199
4	5413	213
5	5627	209

Then we ran the same TSP instance in the search system four more times. Table 2 lists the results of these five trials. The table shows the number of tours found in the matrix E, the sampling average of the best tour found in each of the trials and total computing time consumed by each of the computers. We compared these five best tours and found that they were the same tour; even it bore a different sampling average value in each of trials. This best tour is probably a globally optimal *a prior* tour for the PTSP instance.

Table 2. Results of five trials on the same TSP instance

Trial	Number of Tours in E	Sampling Average for Best Tour	Standard Deviation	Time (seconds)					
				Master	Worker1	Worker2	Worker3	Worker4	Worker5
1	36	4889	201	1007	1578	1549	1561	1468	1593
2	35	4872	209	1200	1563	1600	1589	1564	1465
3	36	4850	195	1157	1480	1533	1474	1593	1572
4	34	4873	199	1004	1550	1472	1543	1569	1530
5	35	4880	202	1058	1461	1486	1542	1473	1609

In another experiment, we used the same TSP instance but generated three different probability arrays P_1, P_2 and P_3. The values in P_1, P_2 and P_3 were generated from the range [0.1, 0.4], [0.3, 0.7] and [0.6, 0.9], respectively. For the probability array P_1, we ran the search system five times. The search system outputted three different best tours in these five trials. It indicates that, when a PTSP instance becomes more stochastic, our search system has more difficulty to find the globally optimal *a priori* tour. Then we ran search system five times for the probability array P_2, the search system outputted the same best tours with different sampling average values. Last we ran search system five times for the probability array P_3, the search system also outputted the same best tour. Obviously, when a PTSP instance is less stochastic or its average probability is 50/50, our search system may be able to find the globally optimal *a priori* tour. The results of this experiment are shown in Table 3.

Table 3. The results of experiment on different probability arrays

Trial	Number of Tours in E	Sampling Average for Best Tour	Standard Deviation	Time (seconds)					
				Master	Worker1	Worker2	Worker3	Worker4	Worker5
P_1									
1	38	1201	275	968	1261	1319	1208	1232	1304
2	37	1180	271	992	1290	1302	1258	1244	1326
3	39	1260	269	985	1239	1211	1215	1309	1269
4	42	1238	267	965	1237	1283	1278	1319	1211
5	39	1189	279	956	1309	1256	1304	1251	1245
P_2									
1	37	5580	166	993	1492	1345	1491	1454	1330
2	37	5606	175	980	1428	1381	1444	1307	1340
3	39	5554	164	1063	1465	1142	1313	1342	1493
4	36	5581	171	1044	1429	1384	1397	1475	1430
5	37	5561	161	995	1432	1369	1481	1330	1382
P_3									
1	34	7047	156	1201	1502	1558	1561	1563	1478
2	35	7049	154	1156	1583	1496	1558	1537	1525
3	33	7012	165	1127	1570	1487	1476	1471	1561
4	36	7009	161	1211	1495	1474	1589	1535	1589
5	37	7108	163	1124	1533	1505	1538	1538	1574

For the PTSP instance with the probability array P_1, we were wondering if we could improve the search quality by increasing the number of realizations in the search process. We set $M_I = 1100$ and $M_{II} = 1850$ and ran the problem instance 10 times. We found five different best tours in the ten trials and five trials outputted the same tour. Then we change $M_I = 3000$ and $M_{II} = 3000$, rand the problem instance 10 times again. We found three different best tours in the ten trials and seven trials outputted the same tour. Then we did the same

procedure using $M_I = 6000$ and $M_{II} = 6000$. This time we found two different best tours and nine trials gave us the same tour. This experiment indicates that we can improve our search by increasing the number of realizations in the search process. Table 4 lists the experiment results.

Table 4. Search results in three different M settings

Setting	Number of Best Tours in 10 Trials	Number of Trials Having the Same Tour	Average Standard Deviation
$M_I = 1100$ $M_{II} = 1850$	5	5	268
$M_I = 3000$ $M_{II} = 3000$	3	7	247
$M_I = 6000$ $M_{II} = 6000$	2	9	226

5 Conclusion

This paper describes a new search algorithm for PTSP, using Monde Carlo simulation and being implemented in a parallel-processing architecture. The search process and simulation for realizations are performed by several parallel processors. If only one processor conducts local search and simulation in a finite time, the search trajectories are trapped in some local valleys and the search system is not really ergodic. Our search system takes a large ensemble of placeMonte Carlo salesmen from different processors to construct the solution attractor; it produces global nature of output. Our search system bears intrinsic parallelism, flexibility and diversity.

Since the primary goal of this paper is to introduce a new search algorithm, demonstrate the general applicability to PTSP, and analyze its search behavior, we didn't spend time on comparison with other algorithms. Work which follows this paper will comprehend the performance analysis of the proposed algorithm and comparison of the algorithm with respect to other PTSP algorithms.

References

1. Bertsimas, D.J., Jaillet, P., Odoni, A.R.: A Priori Optimization. Operations Research 38, 1019–1033 (1990)
2. Jaillet, P.: Probabilistic Traveling Salesman Problems. Ph.D Thesis, Massachusetts Institute of Technology, MA, USA (1985)
3. Jaillet, P.: A Priori Solution of a Traveling Salesman Problem in Which a Random Subset of the Customers Are Visited. Operations Research 36, 929–936 (1988)
4. Bertsimas, D.J.: Probabilistic Combinatorial Optimization Problems. Ph.D Dissertation. Massachusetts Institute of Technology, MA, USA (1988)
5. Bertsimas, D.J., Howell, L.: Further Results on the Probabilistic Traveling Salesman Problem. Journal of Operational Research 65, 68–95 (1993)

6. Jézéquel, A.: Probabilistic Vehicle Routing Problems. Master Thesis, Massachusetts Institute of Technology, MA, USA (1985)
7. Rossi, F., Gavioli, F.: Aspects of Heuristic Methods in the Probabilistic Traveling Salesman Problem. In: Advanced School on Stochastics in Combinatorial Optimization, pp. 214–227. World Scientific, Hackensack (1987)
8. Jaillet, P.: Analysis of Probabilistic Combinatorial Optimization Problems in Euclidean Spaces. Mathematics of Operations Research 18, 51–70 (1993)
9. Birattari, M., Balaprakash, P., Stützle, T., Dorigo, M.: Estimation-Based Local Search for Stochastic Combinatorial Optimization Using Delta Evaluations: A Case Study on the Probabilistic Traveling Salesman Problem. INFORMS Journal on Computing 20, 644–658 (2008)
10. Bianchi, L.: Ant Colony Optimization and Local Search for the Probabilistic Traveling Salesman Problem: A Case Study in Stochastic Combinatorial Optimization. Ph.D Dissertation, Universite Libre de Bruxelles, Brussels, Belgium (2006)
11. Bianchi, L., Campbell, A.M.: Extension of the 2-p-opt and 1-shift Algorithms to the Heterogeneous Probabilistic Traveling Salesman Problem. European Journal of Operational Research 176, 131–144 (2007)
12. Bianchi, L., Gambardella, L.M., Dorigo, M.: An Ant Colony Optimization Approach to the Probabilistic Traveling Salesman Problem. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 883–892. Springer, Heidelberg (2002)
13. Binachi, L., Knowles, J., Bowler, N.: Local Search for the Probabilistic Traveling Salesman Problem: Correction to the 2-p-opt and 1-Shift Algorithms. European Journal of Operational Research 16, 206–219 (2005)
14. Bowler, N.E., Fink, T.M., Ball, R.C.: Characterization of the Probabilistic Traveling Salesman Problem. Physical Review E 68, 1–7 (2003)
15. Branke, J., Guntsch, M.: Solving the Probabilistic TSP with Ant Colony Optimization. Journal of Mathematical Modeling and Algorithms 3, 403–425 (2004)
16. Campbell, A.M.: Aggregation for the Probabilistic Traveling Salesman Problem. Computers and Operations Research 33, 2703–2724 (2006)
17. Liu, Y.-H.: Solving the Probabilistic Traveling Salesman Problem Based on Genetic Algorithm with Queen Selection Scheme. In: Greco, F. (ed.) Traveling Salesman Problem, pp. 157–172. InTech (2008)
18. Liu, Y.-H., Jou, R.-C., Wang, C.-C., Chiu, C.-S.: An Evolutionary Algorithm with Diversified Crossover Operator for the Heterogeneous Probabilistic TSP. In: Torra, V., Narukawa, Y., Yoshida, Y. (eds.) MDAI 2007. LNCS (LNAI), vol. 4617, pp. 351–360. Springer, Heidelberg (2007)
19. Marinakis, Y., Migdalas, M., Pardalos, P.M.: Expanding Neighborhood Search GRASP for the Probabilistic Traveling Salesman Problem. Optimization Letters 2, 351–360 (2008)
20. Marinakis, Y., Marinakis, M.: A Hybrid Multi-Swarm Particle Swarm Optimization Algorithm for the Probabilistic Traveling Salesman Problem. Computers & Operations Research 37, 432–442 (2010)
21. Tang, H., Miller-Hooks, E.: Approximate Procedures of the Probabilistic Traveling Salesperson Problem. Transportation Research Record 1882, 27–36 (2004)
22. Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J.: A Survey on Metaheuristics for Stochastic Combinatorial Optimization. Natural Computing 8, 239–287 (2009)
23. Kleywegt, A.J., Shapiro, A., Homen-de-Mello, T.: The Sample Average Approximation Method for Stochastic Discrete Optimization. SIAM Journal on Optimization 12, 479–502 (2001)

24. Verweij, B., Ahmed, S., Kleywegt, A.J., Nemhauser, G., Shapiro, A.: The Sample Average Approximation Method Applied to Stochastic Routing Problems: A Computational Study. Computational Optimization and Application 24, 289–333 (2003)
25. Balaprakash, P., Pirattari, P., Stützle, T., Dorigo, M.: Adaptive Sample Size and Importance Sampling in Estimation-Based Local Search for the Probabilistic Traveling Salesman Problem. European Journal of Operational Research 199, 98–110 (2009)
26. Balaprakash, P., Pirattari, P., Stützle, T., Dorigo, M.: Estimation-based Metaheuristics of the Probabilistic Traveling Salesman Problem. Computers & Operations Research 37, 1939–1951 (2010)
27. Homen-de-Mello, T.: Variable-Sample Methods for Stochastic Optimization. ACM Transactions on Modeling and Computer Simulation 13, 108–133 (2003)
28. Weyland, D., Bianchi, L., Gambardella, L.M.: New Approximation-Based Local Search Algorithms for the Probabilistic Traveling Salesman Problem. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) EUROCAST 2009. LNCS, vol. 5717, pp. 681–688. Springer, Heidelberg (2009)
29. Aart, E., Lenstra, J.K.: Local Search in Combinatorial Optimization. Princeton University Press, Princeton (2003)
30. Li, W.: Seeking Global Edges for Traveling Salesman Problem in Multi-Start Search. Journal of Global Optimization 51, 515–540 (2011)
31. Li, W.: A Parallel Multi-Start Search Algorithm for Dynamic Traveling Salesman Problem. In: Pardalos, P.M., Rebennack, S. (eds.) SEA 2011. LNCS, vol. 6630, pp. 65–75. Springer, Heidelberg (2011)
32. Li, W., Feng, M.: A Parallel Procedure for Dynamic Multi-Objective TSP. In: Proceedings of 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, pp. 1–8. IEEE Computer Society (2012)
33. Sudman, S.: Applied Sampling. Academic Press, New York (1976)
34. Walson, J.: How to Determine a Sample Size. Penn Cooperative Extension, University Park, PA (2001)

Unsupervised Classifier Based on Heuristic Optimization and Maximum Entropy Principle

Edwin Aldana-Bobadilla and Angel Kuri-Morales

Universidad Nacional Autónoma de México, Mexico City, Mexico
Instituto Técnologico Autónomo de México, México City, Mexico
ealdana@uxmcc2.iimas.unam.mx,
akuri@itam.mx

Abstract. One of the basic endeavors in Pattern Recognition and particularly in Data Mining is the process of determining which unlabeled objects in a set do share interesting properties. This implies a singular process of classification usually denoted as "clustering", where the objects are grouped into k subsets (clusters) in accordance with an appropriate measure of likelihood. Clustering can be considered the most important unsupervised learning problem. The more traditional clustering methods are based on the minimization of a similarity criteria based on a metric or distance. This fact imposes important constraints on the geometry of the clusters found. Since each element in a cluster lies within a radial distance relative to a given center, the shape of the covering or hull of a cluster is hyper-spherical (convex) which sometimes does not encompass adequately the elements that belong to it. For this reason we propose to solve the clustering problem through the optimization of Shannon's Entropy. The optimization of this criterion represents a hard combinatorial problem which disallows the use of traditional optimization techniques, and thus, the use of a very efficient optimization technique is necessary. We consider that Genetic Algorithms are a good alternative. We show that our method allows to obtain successfull results for problems where the clusters have complex spatial arrangements. Such method obtains clusters with non-convex hulls that adequately encompass its elements. We statistically show that our method displays the best performance that can be achieved under the assumption of normal distribution of the elements of the clusters. We also show that this is a good alternative when this assumption is not met.

Keywords: Clustering, Genetic Algorithms, Shannon's Entropy, Bayesian Classifier.

1 Introduction

Pattern recognition is a scientific discipline whose purpose is to describe and classify objects. The descriptive process involves a symbolic representation of

M. Emmerich et al. (eds.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV*, Advances in Intelligent Systems and Computing 227,
DOI: 10.1007/978-3-319-01128-8_2,

these objects called *patterns.* In this sense, the most common representation is through a numerical vector $\boldsymbol{x}$:

$$\boldsymbol{x} = [x_1, x_2, \ldots x_n] \in \Re^n \tag{1}$$

where the n components represent the value of the properties or attributes of an object. Given a pattern set X, there are two ways to attempt the classification: a) *Supervised Approach* and b) *Unsupervised Approach.*

In the supervised approach, $\forall \boldsymbol{x} \in X$ there is a class label $y \in \{1, 2, 3, ..., k\}$. Given a set of class labels Y corresponding to some observed patterns $\boldsymbol{x}$ ("training" patterns), we may postulate a hypothesis about the structure of X that is usually called the *model.* The model is a mathematical generalization that allows us to divide the space of X into k decision regions called *classes.* Given a model M, the class label y of an unobserved (unclassified) pattern $\boldsymbol{x}'$ is given by:

$$y = M(\boldsymbol{x}') \tag{2}$$

On the other hand, the unsupervised approach consists in finding a hypothesis about the structure of X based only on the similarity relationships among its elements. The unsupervised approach does not use prior class information. The similarity relationships allow to divide the space of X into k subsets called *clusters.* A cluster is a collection of elements of X which are "similar" between them and "dissimilar" to the elements belonging to other clusters. Usually the similarity is defined by a *metric* or distance function $d : X \times X \to \Re$.

In this work we discuss a clustering method which does not depend explicitly on minimizing a distance metric and thus, the shape of the clusters is not constrained by hyper-spherical hulls. Clustering is a search process on the space of X that allows us to find the k clusters that satisfy an optimization criteria. Mathematically, any criterion involves an objective function f which must be optimized. Depending on the type of f, there are several methods to find it. Since our clustering method involves an objective function f where its feasible space is, in general, non-convex and very large, a good optimization algorithm is compulsory. With this in mind, we made a comprehensive study [13] which dealt with the relative performance of a set of structurally different GAs and a non-evolutionary algorithm over a wide set of problems. These results allowed us to select the statistically "best" algorithm: the EGA [20]. By using EGA we may be sure that our method will displays high effectiveness for complex arrangements of X.

The paper is organized as follows: In Section 2, we briefly show the results that led us to select the EGA. Then we present the different sets of patterns X that will serve as a the core for our experiments. We use a Bayesian Classifier[2,4,8] as a method of reference because there is theoretical proof that its is optimal given data stemming from normal distributions. In this section we discuss the issues which support our choice. In Section 3 we discuss the main characteristics of our method and the experiments which show that it is the best alternative. In Section 4 we present our general conclusions.

2 Preliminaries

As pointed out above, a "good" optimization algorithm must be selected. We rest on the conclusions of our previous analysis regarding the performance of a set of GAs[13] . Having selected the best GA, we prove the effectiveness of our clustering method by classifying different pattern sets. To this effect, we generated pattern sets where, for each pattern, the class of the objects is known. Hence, the class found by our clustering method may be compared to the true ones. To make the problems non-trivial we selected a non-linearly separable problems. We discuss the process followed to generate these sets. Finally, we resort to a *Bayesian Classifier* [4] in order to show that the results obtained by our method are similar to those obtained with it.

2.1 Choosing the Best Optimization Algorithm

This section is a very brief summary of the most important results found in [13]. A set A of 4 structurally different GAs and a non-evolutionary algorithm (NEA) was selected in order to solve, in principle, an unlimited supply of systematically generated functions in $\Re \times \Re$(called unbiased functions). An extended set of such functions in $\Re \times \Re^2$ and $\Re \times \Re^3$ was generated and solved. Similar behavior of all the GAs in$A$ (within statistical limits) was found. This fact allowed us to hypothesize that the expected behavior of A for functions in $\Re \times \Re^n$ will be similar. As supplement, we tackled a suite of problems (approximately 50) which includes hard unconstrained problems (which traditionally have been used for benchmarking purposes) [19,3] and constrained problems [11]. Lastly, atypical GA-hard functions were analyzed [18,16].

Set of Algorithms. The set A included the following GAs: a)An elitist canonical GA (in what follows referred to as TGA [eliTist GA]) [21], b) A Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation algorithm (CHC algorithm) [5], c) An Eclectic Genetic Algorithm (EGA) [20], d) A Statistical GA (SGA) [23,12] and e) A non-evolutionary algorithm called RMH [17].

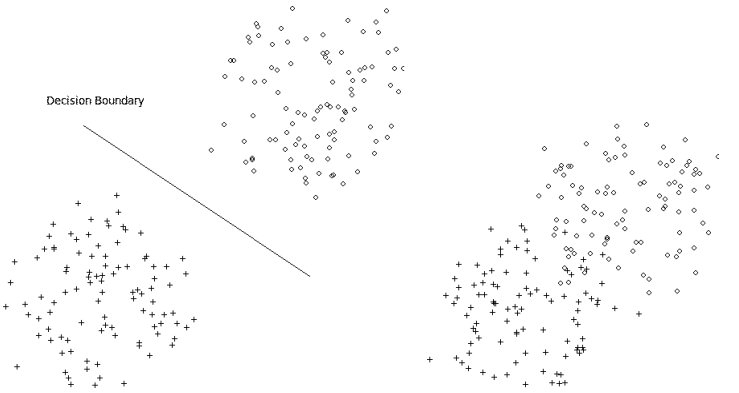
Table 1 shows the relative global performance of all algorithms for the functions mentioned. The best algorithm in the table is EGA.

Table 1. Global Performance

A_i	Unbiased	Suite	Atypical	Global Performance	Relative
EGA	9.64	8.00	4.48	7.37	100.00%
RMH	6.24	0.012	2.04	2.76	37.49%
TGA	1.35	1.16	4.77	2.43	32.91%
SGA	1.33	0.036	3.33	1.57	21.23%
CHC	2.12	0.08	2.10	1.43	19.44%

2.2 The Pattern Set

Given a set of patterns to be classified, the goal of any classification technique is to determine the decision boundary between classes. When these classes are unequivocally separated from each other the problem is separable; otherwise, the problem is non-separable. If the problem is linarly separable, the decision consists of a hyperplane. In Figure 1 we illustrate ths situation.



(a) Linearly Separable patterns (b) Non-linearly separable patterns

Fig. 1. Decision boundary

When there is overlap between classes some classification techniques (e.g. Linear classifiers, Single-Layer Perceptrons [8]) may display unwanted behavior because decision boundaries may be highly irregular . To avoid this problem many techniques has been tried (e.g. Support Vector Machine [9], Multilayer Perceptrons [22]). However, there is no guarantee that any of this methods will perform adequately. Nevertheless, there is a case which allows us to test the appropriateness of our method. Since it has been proven that if the classes are normally distributed, a Bayesian Classifier yields the best possible result (in Section 2.3 we discuss this fact) and the error ratio will be minimized. Thus, the Bayesian Classifier becomes a good method with which to compare any alternative clustering algorithm.

Hence, we generated Gaussian pattern sets considering singular arrangements in which determining the decision boundaries imply non-zero error ratios. Without loss generality we focus on patterns defined in $\Re^2$. We wish to show that the results obtained with our method are close to those obtained with a Bayesian Classifier; in Section 3, the reader will find the generalization of our method for $\Re^n$.

Gaussian Patterns in $\Re^2$. Let X_j be a pattern set defined in $\Re^2$ and $C_i \subset X_j$ a pattern class. A pattern $\boldsymbol{x} = [x_1, x_2] \in C_i$ is drawn from a Gaussian distribution if its joint probability density function (pdf) is given by:

$$f(x_1, x_2) = \frac{1}{2\pi\sigma_{x_1}\sigma_{x_2}\sqrt{1-\rho^2}} e^{\left(-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x_1-\mu_{x_1}}{\sigma_{x_1}}\right)^2 - 2\rho\frac{(x_1-\mu_{x_1})(x_2-\mu_{x_2})}{\sigma_{x_1}\sigma_{x_2}} + \left(\frac{x_2-\mu_{x_2}}{\sigma_{x_2}}\right)^2\right]\right)} \tag{3}$$

where $-1 < \rho < 1$, $-\infty < \mu_{x_1} < \infty$, $-\infty < \mu_{x_2} < \infty$, $\sigma_{x_1} > 0$, $\sigma_{x_2} > 0$. The value ρ is called the correlation coefficient.

To generate a Gaussian pattern $\boldsymbol{x} = [x_1, x_2]$, we use the *acceptance-rejection* method [1,10] which allows us to generate random observations (x_1, x_2) that are drawn from $f(x_1, x_2)$. In this method, a uniformly distributed random point (x_1, x_2, y) is generated and accepted iff $y < f(x_1, x_2)$. In Figure 2.1 we show different pattern sets obtained by applying this method with distinct statistical arguments in (3)

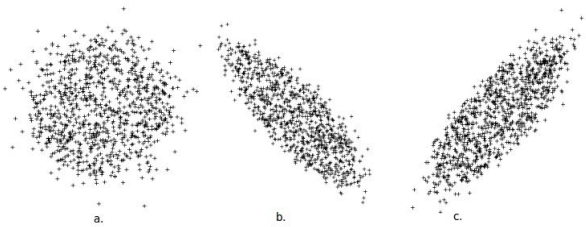


Fig. 2. Different Gaussian pattern sets with $\mu_{x_1} = \mu_{x_2} = 0.5$,$\sigma_{x_1} = \sigma_{x_2} = 0.09$. Each set was generated with different correlation coefficient: a. $\rho = 0.0$, b. $\rho = -0.8$, c. $\rho = 0.8$.

The degrees of freedom in (3) allow us to generate Gaussian pattern sets with varied spatial arrangements. In principle, we analyze pattern sets with the following configurations:

- Sets with disjoint pattern classes.
- Sets with pattern classes that share some elements (partial overlap between classes)
- Sets with pattern classes whose members may share most elements (total overlap).

We proposed these configurations in order to increase gradually the complexity of the clustering problem and analyze systematically the performance of our method. In the following subsections, we make a detailed discussion regarding the generation of sets with these configurations.