2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Parameterized Complexity Analysis and More Effective Construction Methods for ACO Algorithms and the Euclidean Traveling Salesperson Problem

Samadhi Nallaperuma
Evolutionary Computation Group
School of Computer Science
The University of Adelaide
Adelaide, SA 5005, Australia

Andrew M. Sutton
Department of Computer Science
Colorado State University
Fort Collins, CO, USA

Frank Neumann
Evolutionary Computation Group
School of Computer Science
The University of Adelaide
Adelaide, SA 5005, Australia

***Abstract*—We propose a new construction procedure for ant colony optimization (ACO) algorithms working on the Euclidean traveling salesperson problem (TSP) that preserves the ordering on the convex hull of the points in the instance. The procedure is inspired by theoretical analyses for simple evolutionary algorithms that are provably more efficient on instances where the number of inner points of the instance is not too large. We integrate the construction procedure into the well-known Max-Min Ant System (MMAS) and empirically show that it leads to more efficient optimization on instances where the number of inner points is not too high.**

I. Introduction

Meta heuristic approaches such as evolutionary algorithms and ant colony optimization perform well for many combinatorial optimization problems. However, it is hard to understand their working behavior from a mathematical perspective due to the complex random processes that underlie the run of such algorithms. During the last 20 years the computational complexity analysis of evolutionary algorithms has brought in new insights into the working principles of these meta heuristics. Results have been obtained for artificial pseudo-Boolean functions and a wide range of classical combinatorial optimization problems such as shortest paths, minimum spanning trees, maximum matching, covering, cutting and scheduling problems. For a comprehensive presentation we refer the reader to [2, 12].

In the case of ant colony optimization (ACO) a similar path has been followed since 2006. Different studies have been carried out to understand the runtime behavior of ACO algorithms from a mathematical perspective. Initial investigations considered the behavior on simple pseudo-Boolean functions [6, 9, 10]. Later on, classical combinatorial optimization problems such as the minimum spanning tree problem [11], shortest paths problems [18, 19], and the traveling salesperson problem [7] have been investigated.

Although these computational complexity studies are very beneficial for understanding which types of problems can be provably solved in polynomial time by meta heuristic approaches, most of these studies did not have direct implications on the design of new meta heuristic approaches.

The goal of this paper is to show how insights from the computational complexity analysis can be turned into more effective meta heuristics. Recently, the first parameterized computational complexity results have been obtained for evolutionary algorithms and some classical NP-hard combinatorial optimization problems such as vertex cover [8], make span scheduling [21], and the Euclidean traveling salesperson problem [20]. Such studies provide insights into the runtime behavior of meta heuristics in relation to structural parameters of the investigated problem. In [20], it has been shown that simple evolutionary algorithms solve the Euclidean TSP efficiently if the number of inner points of the given instance is not too large. We exploit the structural properties of the Euclidean traveling salesperson problem and their relation to the runtime behavior of ACO algorithms. Our extension of the parameterized complexity analysis of [20] to ACO derives new insights which can be used for the design of more effective ACO algorithms. Based on these insights, we propose new construction procedures for ACO and the Euclidean TSP. We show that this approach leads to an XP-algorithm for the Euclidean TSP. Furthermore, we integrate this construction procedure into the well-known Max-Min Ant System (MMAS) [17] and show in our experimental studies that it leads to more efficient optimization on instances where the number of inner points is not too high.

The structure of the paper is as follows. Section II introduces the problem formally and discusses solutions based on parameterizations of ant algorithms. Section III explains our approach to formulate the XP version of the ACO algorithm to solve TSP. Section IV describes complete ACO algorithms including the introduced tour construction and new local search operator. Section V presents some experimental results of the proposed algorithms compared with classical MMAS. Finally, we finish with concluding remarks.

II. Preliminaries

The traveling salesperson problem (TSP) is one of the most famous NP-hard combinatorial optimization problems. Given a set of n cities $\{1, \ldots, n\}$ and a distance matrix $d = (d_{i,j})$, $1 \leq i, j \leq n$, the goal is to compute a minimal length Hamiltonian cycle (a tour that visits each city exactly once and returns to the origin). In the case of complete graphs, Hamiltonian cycles can be represented as permutations of the n cities. For a given

978-1-4799-0454-9/13/$31.00 ©2013 IEEE

permutation $\pi = (x_1, \ldots, x_n)$ we denote by

$$c(\pi) = d_{x_n,x_1} + \sum_{i=1}^{n-1} d_{x_i,x_{i+1}}$$

the cost of the tour π.

In our study we consider a well-known ACO algorithm called Max-Min Ant System (MMAS) [17]. The general structure of ACO procedures is outlined in Algorithm 1. Individual solution tours are constructed at each iteration by a set of artificial ants. These tours are built by visiting each node in a tour sequentially according to a probabilistic formula that takes into account instance-specific heuristic information and pheromone trails. This probabilistic formula, called the random proportional rule, specifies a transition probability p_{ij} that an ant currently visiting node i selects node j next in its tour. This probability is defined as

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\left(\sum_{h \in N_k} [\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}\right)} \quad (1)$$

Here N_k represents the set of nodes that have not yet been visited by ant k, $[\tau_{ih}]$ and $[\eta_{ih}]$ represent pheromone trail intensity and heuristic information respectively. The parameters α and β adjust the influence of the pheromone and the heuristic on the selection decision.

In the MMAS, the pheromone update is performed after the tour construction of ants according to $\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau^b_{ij}$, where ρ denotes the evaporation rate and

$$\Delta\tau^b_{ij} = \begin{cases} 1/C_b & \text{if } (i,j) \text{ belongs to the best-so-far tour;} \\ 0 & \text{otherwise.} \end{cases}$$

Here C_b is the cost of the best-so-far tour (the iteration-best tour is also sometimes used). Additionally, local search can be applied on these constructed solutions to improve them further. Pheromone trails in MMAS are bounded between maximum and minimum limits τ_{max} and τ_{min}. At the beginning of a run, the best solution corresponds to the one found by the nearest neighbor heuristic. A detailed description and analysis of this algorithm on TSP can be found in the textbook of Dorigo and Stützle (Chapter 3) [4].

Algorithm 1: Outline of Ant Colony Optimization (ACO)

```
Set parameters; Initialize pheromone trails;
while termination condition not met do
    Construct tour ;
    Apply local search (Optional);
    Update Pheromones ;
```

We propose new construction procedures for ACO and the TSP which are based on insights from parameterized complexity analysis. Parameterized complexity theory proposed by Downey and Fellows [5] is an extension to traditional complexity theory that incorporates additional parameters into running time analysis. The analysis on hard algorithmic problems is decomposed into these parameters of the problem input. This approach illuminates the relationship between the hardness and the problem structure by isolating the exponential complexity into few parameters. When randomized algorithms such as ACO algorithms are considered, the runtime is often expressed as a random variable T that measures the number of steps the algorithm takes to decide a parameterized problem. A randomized algorithm with *expected* optimization time $E(T) \leq g(k) * n^{O(1)}$ (respectively , $E(T) \leq n^{g(k)}$) is a randomized fixed parameter tractable algorithm (respectively, XP-algorithm) for the corresponding parameterization k [20].

A natural parameterization for the Euclidean TSP is based on the points that lie on the interior of the convex hull of an instance [3, 20]. Deĭneko et. al [3] describe the parameterized problem as follows. Let $V \subseteq \mathbb{R}^2$ be a finite point set in the Euclidean plane. The convex hull of V is the smallest convex set containing V. A point $p \in V$ is called an inner point if p lies in the interior of the convex hull. The set of inner points of V is denoted by $Inn(V)$. A point $p \in V$ is called an outer point if it is not an inner point. Outer points are denoted by $Out(V)$. Hence $V = Inn(V) \cup Out(V)$ and $Inn(V) \cap Out(V) = \emptyset$. Throughout this paper, we denote by $n := |V|$ the size of an instance of TSP and $k := |Inn(V)|$ the number of inner points in an instance.

Let $v_1^{out} \in Out(V)$ be an arbitrary outer point. We denote by $(v_1^{out}, v_2^{out}, \ldots, v_{n-k}^{out})$ the sequence of outer points clockwise along the convex hull when starting with v_1^{out} (for an illustration see Figure 2). We call such a sequence a cycling ordering of the points in $Out(V)$.

III. New Construction Procedures

In this section, we introduce new construction procedures based on parameterized complexity results. Furthermore, we analyze the new construction methods with respect to their theoretical properties. Our goal is to study the structure of the underlying search space given by the different construction methods. For all investigations carried out in this section, we set $\alpha = \beta = 0$ which implies that ACO algorithms construct solutions without taking pheromone or heuristic information into account.

When a point set V of size n is supplied to an algorithm as input, the convex hull of V can be computed in $O(n \log n)$ time. Then, as outlined in Algorithm 2, the original loop of ACO runs until the termination condition is met. This termination condition could be anything similar to classical ACO. The only difference from classical ACO shown in Algorithm 1, is that the selection of the next city during construction preserves convex hull order in the constructed tour. In this section, we carry out a rigorous analysis of the tour construction procedure.

Algorithm 2: Parameterized ACO for Euclidean TSP

```
Set parameters; Initialize pheromone trails;
while termination condition not met do
    Construct tour preserving convex hull order;
    Apply local search (Optional);
    Update Pheromones ;
```

We say a tour x is *hull-respecting* if the subsequence of points in x belonging to $Out(V)$ appear in the same order

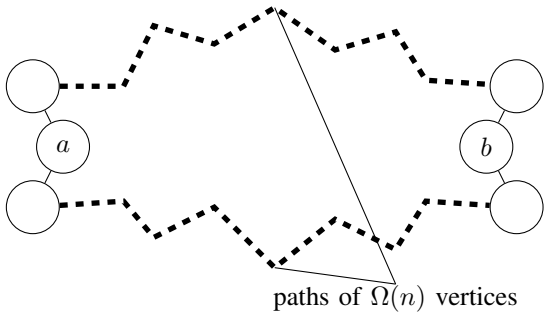

Fig. 1. Example instance with two interior points

as some cyclic ordering of $Out(V)$. The following theorem follows immediately from the proof of Theorem 2 in [13].

Theorem 1: If $x^\star$ is optimal, then $x^\star$ is hull-respecting.

Furthermore, if x is a 2-opt local optimum, it must also be hull-respecting since the resulting tour must not contain any intersecting edges [20].

Therefore, it might be useful in practice to constrain the construction of tours in such a way that only hull-respecting tours are generated. We now consider a few ways of achieving this.

Function Construct1$(V, \tau, \eta, \alpha, \beta)$

$x_1 \leftarrow v_1^{out}$;
$h \leftarrow 1$;
for $i = 1$ **to** $n - 1$ **do**
  $V' \leftarrow \{v \in V : v \text{ is unvisited}\}$;
  $E' \leftarrow \{(x_i, v) : v \in V' \setminus Out(V)\}$;
  if $h \leq |Out(V)|$ **then** $E' \leftarrow E' \cup \{(x_i, v_h^{out})\}$;
  $R \leftarrow \sum_{e \in E'} \tau(e)^\alpha \eta(e)^\beta$;
  Choose one edge $e = (x_i, v)$ from E' with probability $\left(\tau(e)^\alpha \eta(e)^\beta\right)/R$;
  if $v \in Out(V)$ **then** $h \leftarrow h + 1$;
  $x_{i+1} \leftarrow v$;
return $\pi = (x_1, \ldots, x_n)$;

Function Construct1 constructs tours using the heuristic and the pheromone values, but has the added mechanism of ensuring points that lie on the convex hull are always placed in the correct order. However, the procedure is poorly biased, since it does not generate such tours in a uniform manner. Because of this effect, the runtime can be arbitrarily bad, even with a few interior points.

We consider the TSP instance class T^w defined as follows. Let V be a set of n points in the Euclidean plane with 2 interior points, labeled a and b, such that, in every optimal Hamiltonian tour, a and b are connected by two paths of $\Omega(n)$ vertices. An example is shown in Figure 1.

Theorem 2: Let $\alpha = \beta = 0$. The expected time until ACO using the tour construction function Construct1 finds an optimal tour for any instance of T^w is bounded below by $\Omega(e^n)$.

Proof: Without loss of generality, suppose that the construction procedure places vertex a before b (if it selects b first, the same argument can be made transposing a and b). Furthermore, suppose that it has already constructed a partial optimal solution from x_1 to a. In order to construct a complete optimal solution, the procedure must correctly place $\Omega(n)$ vertices correctly before selecting b. The probability that the construction procedure places any unplaced point at a given position is at most $1/(k+1)$. So in this case, the probability that the construction procedure places vertex b correctly is at most

$$\left(1 - \frac{1}{k+1}\right)^{\Omega(n)} \frac{1}{k+1} \leq e^{-cn}$$

for a positive constant c. Thus the waiting time until the construction at least places vertex b correctly is at least $\Omega(e^n)$.

A similar argument holds if b is placed before a. It thus follows that the expected time to wait until the construction procedure builds any optimal tour for such an instance is at least exponential. ■

We now make a small modification to Function Construct1 that ensures all tours with the points on the convex hull in the correct order are generated uniformly. This improves the runtime with respect to k, and yields a randomized XP-algorithm for Deĭneko's parameterization of Euclidean TSP.

Function Construct2$(V, \tau, \eta, \alpha, \beta)$

$x_1 \leftarrow v_1 \in V$ uniformly at random;
$h \leftarrow 0$;
if $x_1 \in Out(V)$ **then**
  $h \leftarrow 1$;
for $i = 1$ **to** $n - 1$ **do**
  $V' \leftarrow \{v \in V : v \text{ is unvisited}\}$;
  $E' \leftarrow \{(x_i, v) : v \in V' \setminus Out(V)\}$;
  $R \leftarrow \sum_{e \in E'} \tau(e)^\alpha \eta(e)^\beta$;
  Choose a real number r uniformly at random from $(0, 1)$;
  if $r < (n - k - h)/(n - i)$ **then**
    $x_{i+1} \leftarrow v_{h+1}^{out}$;
    $h \leftarrow h + 1$;
  else
    Choose one edge $e = (x_i, v)$ from E' with probability $\left(\tau(e)^\alpha \eta(e)^\beta\right)/R$;
    $x_{i+1} \leftarrow v$;
return $\pi = (x_1, \ldots, x_n)$;

Theorem 3: Let $\alpha = \beta = 0$. ACO using the tour construction function Construct2 is a randomized XP-algorithm for the interior-point parameterization of Euclidean TSP. In particular, for any instance with n total points and k interior points, the construction procedure generates an optimal solution after at most $O(n^k)$ steps in expectation.

Proof: Let $x^\star$ be an optimal tour starting with the node x_1 chosen by Construct2. During the construction procedure of the tour, the probability that x_i, $2 \leq i \leq n$, is assigned the vertex corresponding to $x_i^\star$ conditioned on the event that $x_j = x_j^\star$ for all $j < i$ is

$$p_i = \begin{cases} (n - k - h)/(n - i) & \text{if } x_i^\star \in Out(V); \\ 1/(n - i) & \text{otherwise.} \end{cases}$$

where h is the number of vertices in $Out(V)$ that have already been chosen.

Therefore, the probability that any run through the construction procedure produces exactly $x^\star$ is at least

$$\prod_{i=1}^{n} p_i \geq \frac{(n-k-1)\cdot(n-k-2)\cdots 3\cdot 2\cdot 1}{(n-1)\cdot(n-2)\cdots 3\cdot 2\cdot 1} = \frac{(n-k-1)!}{(n-1)!}.$$

Thus, the expected number of calls to Function Construct2 until $x^\star$ is generated is at most

$$\frac{(n-1)!}{(n-k-1)!} = O(n^k)$$

which completes the proof. ■

The previous theorem implies that all instances in T^w that were solved in expected exponential time by Construct1 can be solved by Construct2 in expected time $O(n^2)$.

These theoretical results suggest that a principled construction procedure can constrain the search to a subset of the search space that contain the optimal tours. If an instance has fewer inner points, this subset may be manageable. We now consider modified ACO algorithms that employ this principled construction procedure, and we compare their performance to the original MMAS.

In the following, we present a small experimental study of the different construction procedures. To isolate the effects of construction, we set $\alpha = \beta = 0$. This eliminates the effect of pheromone values and heuristic information, and allows us to measure how each procedure is sampling the search space. We measure the solution quality obtained within 1000 generations and compare it to the value of an optimal solution (OPT) computed by the exact TSP solver Concorde [1]. Table I shows the results of the four algorithms on TSP instances with 25 cities. These instances have an increasing number of inner points (1-5.tsp inner points 5% of total points, 6-10.tsp 10% and 11-15.tsp 20%) within a reasonable range to support our claim for "few" inner points. For all instances, employing the construction procedure used by classical MMAS (c.f., the column labeled "original") results in tours with lengths that are far from optimal. In contrast to this, the ACO algorithm employing our new construction procedures (Construct 1 and 2) find an optimal tour in most cases. For the set of instances with 20% of inner points (11-15.tsp), the theoretical worst-case runtime bound given in Theorem 3 for Construct2 is already much larger than 1000, and the theoretical worst-case bound for Construct1 is even higher. Nevertheless, Construct2 manages to achieve near-optimal tour lengths for those instances after 1000 iterations.

instance	original	Construct1	Construct2	OPT
1.tsp	141698	49621	49621	49621
2.tsp	134904	50866	50866	50866
3.tsp	152281	57851	57851	57851
4.tsp	125082	51200	51200	51200
5.tsp	139939	58057	58057	58057
6.tsp	137554	60250	60250	60250
7.tsp	132794	53876	53784	53784
8.tsp	131903	60771	60771	60771
9.tsp	134368	63079	55049	55049
10.tsp	137953	63921	59678	58886
11.tsp	132502	68073	68411	66592
12.tsp	136393	68407	64737	60613
13.tsp	129795	65360	61665	54074
14.tsp	132828	70113	63642	58705
15.tsp	147072	70564	68383	67314

TABLE I. RESULTS FOR DIFFERENT CONSTRUCTION PROCEDURES WITH $\alpha = \beta = 0$.

IV. NEW ACO ALGORITHMS

In this section, we propose new ACO approaches based on the theoretical investigations carried out in the previous section. We design two improved Max-Min versions by integrating nearest neighbor lists as in original MMAS to the two tour construction procedures described above.

The first algorithm is outlined in Function XPMMAS1 and addresses the poor bias of the tour construction procedure in Construct1 by enforcing a higher probability of choosing outer points over inner points in each step. In contrast to the equal probability of choosing points in Construct1 explained in the previous section, here an outer point is selected with $(n-k-h)$ times higher probability than the probability of choosing an inner point. The selection of the next point is based on the nearest-neighbor list, as in original MMAS, but with the additional constraint of convex hull order preservation. In other words, each next point is selected from unvisited nodes in the nearest neighbor list, subject to preserving ordering on the convex hull within the tour under construction. The remainder of the algorithm, such as pheromone update and initialization, are the same as in the original MMAS.

The second algorithm outlined in Function XPMMAS2 operates by choosing an outer point with the appropriate probability and then, if the outer point was not chosen, choosing an inner point from the nearest neighbor list according to the random proportional rule, i.e., Equation (1). This implementation employs the unbiased tour construction outlined in Construct2 with the addition of the nearest neighbor list functionality.

We can observe the difference of the two proposed variants in the phase where the decision of the next point is made. In the first version, any available point is selected based on the transition probabilities determined by pheromone values, heuristic information and whether or not the node lies on the convex hull. In the second version, the probability that the next outer point is chosen only depends on k and the number of outer points h chosen so far. If an outer point is not chosen, an inner point will be chosen according to the transition probabilities determined by pheromone values and heuristic information. Both implementations are integrated into the standard MMAS implementation [16] with the described modifications in the tour construction.

A. Local Search

Classical ACO algorithms are usually combined with local search to improve the runtime and the quality of the solution. For TSP this is often achieved using 2-opt or 3-opt local search operators. Hence, for our XP version of MMAS, we have introduced a local search operator which preserves the

Function XPMMAS1($V, \tau, \eta, \alpha, \beta$)

$x_1 \leftarrow v_1 \in V$ uniformly at random;
$h \leftarrow 0$;
if $x_1 \in Out(V)$ **then**
  $h \leftarrow 1$;
for $i = 1$ **to** $n - 1$ **do**
  $E' \leftarrow \{(x_i, v) : v \in V \setminus Out(V),\ v$ is unvisited$\}$;
  if $h \leq |Out(V)|$ **then** $E' \leftarrow E' \cup \{(x_i, v_h^{out})\}$;
  $NN \leftarrow$ nearest neighbor list of x_i
  $NN' \leftarrow \{(x_i, v) : v \in NN \setminus Out(V),\ v$ is unvisited$\}$;
  if $h \leq |Out(V)|$ **then** $NN' \leftarrow NN' \cup \{(x_i, v_h^{out})\}$;
  $R \leftarrow \sum_{e \in E'} \tau(e)^{\alpha} \eta(e)^{\beta}$;
  $RN \leftarrow \sum_{e \in NN'} \tau(e)^{\alpha} \eta(e)^{\beta}$;
  Choose one edge $e = (x_i, v)$ **if** $NN'! = empty$ **then**
    from NN' with probability
    if $v \in Out(V)$ **then**
      $(n - k - h)/(n - i) * \left(\tau(e)^{\alpha} \eta(e)^{\beta}\right)/RN$;
    else
      $1/(n - i) * \left(\tau(e)^{\alpha} \eta(e)^{\beta}\right)/RN$;
  else
    from E' with probability **if** $v \in Out(V)$ **then**
      $(n - k - h)/(n - i) * \left(\tau(e)^{\alpha} \eta(e)^{\beta}\right)/R$;
    else
      $1/(n - i) * \left(\tau(e)^{\alpha} \eta(e)^{\beta}\right)/R$;
  $x_{i+1} \leftarrow v$;
  if $v \in Out(V)$ **then** $h \leftarrow h + 1$;
return $\pi = (x_1, \ldots, x_n)$;

Function XPMMAS2($V, \tau, \eta, \alpha, \beta$)

$x_1 \leftarrow v_1 \in V$ uniformly at random;
$h \leftarrow 0$;
if $x_1 \in Out(V)$ **then**
  $h \leftarrow 1$;
for $i = 1$ **to** $n - 1$ **do**
  $E' \leftarrow \{(x_i, v) : v \in V \setminus Out(V),\ v$ is unvisited$\}$;
  $NN \leftarrow$ nearest neighbor list of x_i
  $NN' \leftarrow \{(x_i, v) : v \in NN \setminus Out(V),\ v$ is unvisited$\}$;
  $R \leftarrow \sum_{e \in E'} \tau(e)^{\alpha} \eta(e)^{\beta}$;
  $RN \leftarrow \sum_{e \in NN'} \tau(e)^{\alpha} \eta(e)^{\beta}$;
  Choose a real number r uniformly at random from $(0, 1)$;
  if $r < (n - k - h)/(n - i)$ **then**
    $x_{i+1} \leftarrow v_{h+1}^{out}$;
    $h \leftarrow h + 1$;
  else
    if $NN'! = empty$ **then**
      Choose one edge $e = (x_i, v)$ from NN' with probability $\left(\tau(e)^{\alpha} \eta(e)^{\beta}\right)/RN$;
      $x_{i+1} \leftarrow v$;
    else
      Choose one edge $e = (x_i, v)$ from E' with probability $\left(\tau(e)^{\alpha} \eta(e)^{\beta}\right)/R$;
      $x_{i+1} \leftarrow v$;
return $\pi = (x_1, \ldots, x_n)$;

convex hull order while improving the overall solution using local search. We cannot use 2- or 3-opt because such operators will destroy convex hull order once the cities in between the selected points are inverted. Therefore, we propose a new kind of operator that only reorders inner points, and thus preserves order along the convex hull. Our approach is inspired by the jump operator introduced in the genetic algorithm domain [15]. In this operator, an inner point can be shifted to any other position of the permutation, shifting the remaining elements such that their order is preserved. The jump operator is applied iteratively, accepting improving tours until a local optimum is found. The procedure is outlined in Algorithm 3.

Algorithm 3: XP-jump

$V_{inn} \leftarrow \{v \in V \setminus Out(V)\}$;
$V_{inn} :=$ Permute (V_{inn});
while *improvement* **do**
  for *each* $a \in V_{inn}$ **do**
    $a_1 := Tour[Position[a + 1]]$;
    $a_2 := Tour[Position[a - 1]]$;
    for *each* $b_2 \in V$ **do**
      $b_1 := Tour[Position[b_2 - 1]]$;
      gain $:= -d[a_2, a] - d[a_1, a] + d[a_2, a_1] + d[b_1, a] + d[b_2, a] - d[b_1, b_2]$;
      if *gain* < 0 **then**
        shift cities between a and b_2 to front or back based on the jump direction;
        a jump to position of b_2 in the tour;
        restart search;

V. Experimental Investigations

We conducted experiments to compare the new algorithms with the classical MMAS. We consider three different ACO versions with α and β and other ACO parameters set to their default values ($\alpha = 1$, $\beta = 2$, $\rho = 0.02$, $q\_0 = 0.0$ etc.) as in the original ACO package [16]. We compare the XPMMAS1 and XPMMAS2 variants using XP-jump local search to MMAS using 2-opt as the local search operator. Thus, except for the construction procedure and the local search operator, the implementation of the new algorithms and their running conditions are the same as in classical MMAS. We consider several Euclidean 2D TSP instances with various sizes and percentages of inner points. The instance generation process is described in detail in the next section.

We calculated the minimum tour length found within a fixed number of iterations for each algorithm on each instance. We repeated each experiment 10 times on each instance of the set. We then performed a one-tailed Wilcoxon signed rank test [22] to determine whether the proposed construction procedures perform better than MMAS with statistical significance. This test is appropriate for paired random variables and does not make any assumptions on the underlying distribution. We report the resulting rank sum (W) and confidence (p) values. We performed the experiments on an SGI Altix XE 1300 cluster with 70 nodes where each node has 2.66 GHz Intel Clovertown quad core processors, 8GB RAM and a 250GB

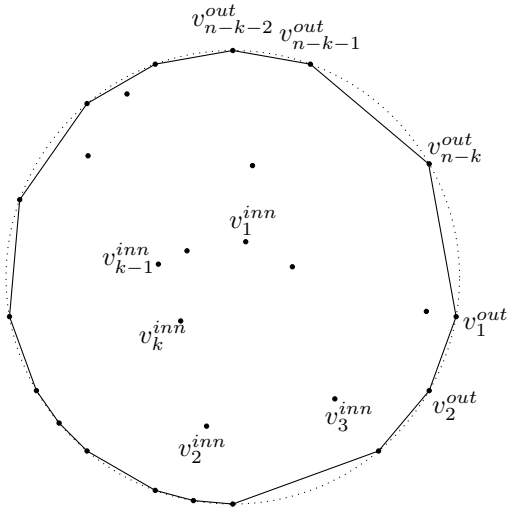

Fig. 2. Instance generation procedure.

SATA drive. We also used the R statistical package [14] to carry out statistical tests.

A. Parameterized Instances

We generated a set of one hundred instances per each category representing instance size (25, 50, 100 and 200) and inner point percentage (5%, 10%, 20%, 30% and 40%). Our goal is to test the impact of the number of inner points. We generate the outer points on a convex hull first and then add the required number of inner points. We generate such sets in order to tightly control the percentage of inner points to ultimately assess its impact on algorithm performance. Such control is not available in current benchmark instances.

To construct each instance, we first create a circle with a given radius and then randomly generate outer points on the circumference by producing a random angle and deriving x and y coordinates from the angle from the y axis. This process continues until we get the required number of outer points to compliment our required number of inner points. We then compute the convex hull formed by the generated points and subsequently insert random points inside the hull. Here, we accept a randomly generated inner point only if it falls inside the convex hull. The instance generation is illustrated in Figure 2.

B. Solution Quality Results

We compare the solution quality obtained after 10000 iterations for the three different algorithms. For each instance size and each inner node percentage, we created a data set consisting of 100 instances using the procedure outlined in the previous section. The average minimum tour length values obtained for each data set are displayed in Table III. To evaluate for statistical significance, the Wilcoxon signed rank test is used on each of these data sets. The test results are presented in Table II. The data set is indicated in the first column, represented with the inner point percentage preceded by the instance size.

For small instance sizes, all three algorithms have similar results while our improved version XPMMAS1 has achieved the best results and the classical MMAS and XPMMAS2 perform slightly worse. For example, for the instance set representing the smallest size and for the least number of inner points (25_5%), the three algorithms have obtained similar values for the 100 TSP instances. This is indicated in comparatively smaller rank sum values for the three test categories resultant from Wilcoxon signed rank test (see Table II). For larger instance sizes, XPMMAS1 has obtained strictly better results than the other two. For example, as shown in Figure 3 XPMMAS1 has achieved smaller tour length values than MMAS for all 100 instances.

The results of the statistical tests provide more insights into the difference of algorithm performance. The first test (*test1*) compares the solution quality of XPMMAS1 with classical MMAS. Based on the observations of the raw results of the experiment, we have built our hypotheses for the rank tests. Hence, for *test1*, we have set the alternative hypothesis that XPMMAS1 performs better (has smaller tour length values) than classical MMAS in terms of solution quality. Hence, the null hypothesis would be that XPMMAS1 does not perform better than MMAS. Throughout this section the symbol "$<$" stands for having smaller tour length values (hence better performance) and "$>$" for larger values. Similarly, we have set the hypotheses that classical MMAS performs better than XPMMAS2 (*test2*) and XPMMAS1 performs better than XPMMAS2 (*test3*).

For all data sets and *test1*, XPMMAS1 has obtained better tour length values than classical MMAS significantly ($p < 0.001$). In such cases, with at least 99.9% confidence we reject the null hypothesis (that XPMMAS1 does not perform better than MMAS) in favor of the alternative hypothesis. The W-values indicate positive rank sums of the difference of two pairs (MMAS and XPMMAS1). Greater values mean a higher difference in algorithm performance. The term positive indicates that we consider the ranks only where MMAS $>$ XPMMAS1 holds to sum up the W-value.

We also observe evidence for a favorable effect in the construction procedure on many cases. Furthermore, the rank sum values increase with the instance size due to the widening gap of the results of the two algorithms. This shows the scalability of our approaches compared to the classical MMAS.

The results for *test2* and *test3* are similar. Both tests report larger tour length values for XPMMAS2 than MMAS and XPMMAS1. Thus, we have evidence that XPMMAS1 usually tends to outperform XPMMAS2 with statistical significance. Nevertheless, in some cases of *test2* we have observed zero rank sums and p-values of 1 suggesting XPMMAS2 performs as well as classical MMAS. This is more apparent in the tests for the instances with fewer inner points. For *test3*, and instance sizes of at least 50 there are large rank sums and confidence values, showing that XPMMAS1 outperforms XPMMAS2 in almost all cases.

In the previous tests, we did not observe a significant performance variation as we varied the percentage of inner points except for a few cases of *test2*. We assume this may be because the local search operator is performing so well that it obscures the effect of the construction procedure. Therefore,

instance	test1		test2		test3	
	W	p-value	W	p-value	W	p-value
25_5%	820	9.095e-13	0	1	0	1
25_10%	2016	2.2e-16	17	1	105	6.104e-05
25_20%	2145	2.2e-16	0	1	276	1.192e-07
25_30%	2145	2.2e-16	253.5	1	780	1.819e-12
25_40%	1326	4.441e-16	254	0.9997	465	9.313e-10
50_5%	4753	2.2e-16	0	1	1275	8.882e-16
50_10%	4753	2.2e-16	331	1	2346	2.2e-16
50_20%	4851	2.2e-16	1203	0.989	4851	2.2e-16
50_30%	4465	2.2e-16	2171.5	1.116e-08	4560	2.2e-16
50_40%	4005	2.2e-16	2695	2.2e-16	4465	2.2e-16
100_5%	5050	2.2e-16	217	1	4005	2.2e-16
100_10%	5050	2.2e-16	1021	0.002459	5050	2.2e-16
100_20%	5050	2.2e-16	1826	2.2e-16	5050	2.2e-16
100_30%	5050	2.2e-16	3570	2.2e-16	5050	2.2e-16
100_40%	4950	2.2e-16	4095	2.2e-16	4950	2.2e-16
200_5%	5050	2.2e-16	1224	7.274e-08	5050	2.2e-16
200_10%	5050	2.2e-16	1418	1.554e-15	5050	2.2e-16
200_20%	5050	2.2e-16	2415	2.2e-16	5050	2.2e-16
200_30%	5050	2.2e-16	2690	2.2e-16	4950	2.2e-16
200_40%	4950	2.2e-16	4656	2.2e-16	4950	2.2e-16

TABLE II. RESULTS OF WILCOXON SIGNED RANK TESTS FOR SOLUTION QUALITY (MINIMUM TOUR LENGTH) WITHIN 10000 ITERATIONS FOR XPMMAS1 < MMAS (*test1*), MMAS < XPMMAS2 (*test2*) AND XPMMAS1 < XPMMAS2 (*test3*), POSITIVE RANK SUMS (W) AND CONFIDENCE (p) VALUES ARE DISPLAYED ACCORDINGLY

instance	MMAS	XPMMAS1	XPMMAS2
25_5%	53281	52949	52949
25_10%	57345	56504	56700
25_20%	60715	59836	60051
25_30%	64157	63443	63825
25_40%	64977	64570	64835
50_5%	61650	59151	60262
50_10%	66314	63713	65163
50_20%	74131	71752	74084
50_30%	79522	77778	80667
50_40%	82648	81485	84101
100_5%	69786	65097	68649
100_10%	79358	75257	79909
100_20%	90687	87819	91715
100_30%	98896	97026	100418
100_40%	105491	104222	107225
200_5%	81584	76196	82387
200_10%	93843	89843	94659
200_20%	110731	108221	111281
200_30%	123343	122072	123958
200_40%	133442	132408	134521

TABLE III. AVERAGE MINIMUM TOUR LENGTH VALUES (ROUNDED TO THE NEAREST WHOLE NUMBER) OBTAINED BY MMAS, XPMMAS1 AND XPMMAS2 WITHIN 10000 ITERATIONS FOR EACH SET OF 100 INSTANCES HAVING SIZES 25, 50, 100 AND 200 AND INNER POINT PERCENTAGES 5%, 10%, 20%, 30% AND 40%

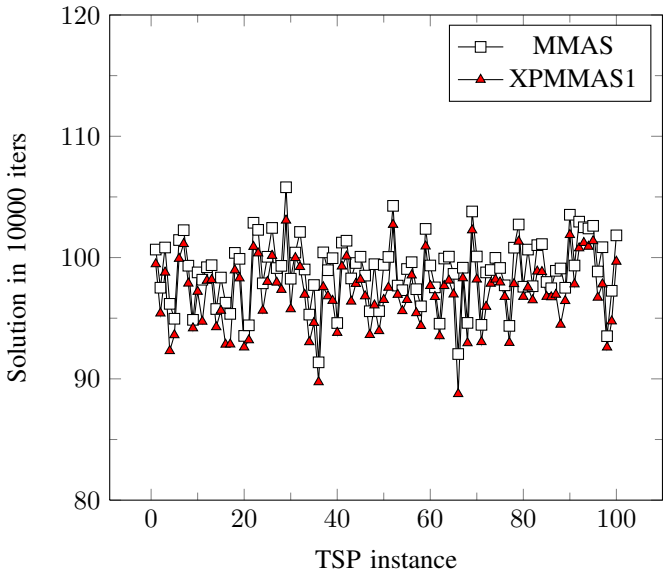

Fig. 3. Comparison of solution quality of classical MMAS and the new version (XPMMAS1) obtained within 10000 iterations for parameters $\alpha = 1$ and $\beta = 2$ for 100 instances of size 100 and inner point percentage 30

we have run another test to determine this effect by removing local search (see Table IV). The respective average values of the data sets are presented in Table V. These results show that when the inner point percentage increases to 40%, our algorithms (without local search) start to perform more poorly. This is expected, since the construction procedures are leveraging the constraints on the convex hull, and we expect the performance to degrade as the number of inner points becomes large.

VI. CONCLUSION AND FUTURE WORK

Parameterized complexity analysis of heuristics enables one to bring in structural insights into the design of these methods. We presented an enhanced ant colony optimization approach for Euclidean TSP based on parameterized results taking into account the number of inner points. For our new construction procedure we derived an upper bound of $O(n^k)$ on the expected runtime if the algorithm samples the search space randomly. Using the new construction procedures in the classical MMAS approach, we obtained more effective ACO

instance	test1		test2		test3	
	W	p-value	W	p-value	W	p-value
25_5%	0	1	0	1	0	1
25_10%	0	1	3741	2.2e-16	3741	2.2e-16
25_20%	0	1	4753	2.2e-16	4753	2.2e-16
25_30%	1	1	5050	2.2e-16	5050	2.2e-16
25_40%	15	0.0625	4950	2.2e-16	4950	2.2e-16

TABLE IV. RESULTS OF WILCOXON SIGNED RANK TESTS FOR SOLUTION QUALITY WITHIN 10000 ITERATIONS WITHOUT LOCAL SEARCH FOR XPMMAS1 > MMAS (*test1*), XPMMAS2 > MMAS (*test2*) AND XPMMAS2 > XPMMAS1 (*test3*), POSITIVE RANK SUMS (W) AND CONFIDENCE (P) VALUES ARE DISPLAYED ACCORDINGLY

instance	MMAS	XPMMAS1	XPMMAS2
25_5%	53281	53281	53281
25_10%	57345	57345	64038
25_20%	60716	60715	68667
25_30%	64157	64157	72857
25_40%	64977	64986	74666

TABLE V. AVERAGE MINIMUM TOUR LENGTH VALUES (ROUNDED TO THE NEAREST WHOLE NUMBER) OBTAINED BY MMAS, XPMMAS1 AND XPMMAS2 WITHIN 10000 ITERATIONS FOR EACH SET OF 100 INSTANCES HAVING SIZES 25, 50, 100 AND 200 AND INNER POINT PERCENTAGES 5%, 10%, 20%, 30% AND 40%

algorithms. Experimental results suggest that our new methods outperform the classical MMAS approach on instances up to a reasonable (roughly 40% of the number of total points) number of inner points. Future work will be concentrated on expanding the experiments to massive scale to test the validity of current results on extremely large instances and improving the current performance of new methods further by integrating new parameterizations that have a complementary effect on the current one, such that the new methods will still outperform general MMAS even when the number of inner points is high.

REFERENCES

[1] D. Applegate, W. J. Cook, S. Dash, and A. Rohe. Solution of a min-max vehicle routing problem. *INFORMS Journal on Computing*, 14(2):132–143, 2002.

[2] A. Auger and B. Doerr, editors. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific, 2011.

[3] V. G. Deĭneko, M. Hoffman, Y. Okamoto, and G. J. Woeginger. The traveling salesman problem with few inner points. *Operations Research Letters*, 34:106–110, 2006.

[4] M. Dorigo and T. Stützle. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA, 2004.

[5] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999. 530 pp.

[6] W. J. Gutjahr. Mathematical runtime analysis of ACO algorithms: Survey on an emerging issue. *Swarm Intelligence*, 1:59–79, 2007.

[7] T. Kötzing, F. Neumann, H. Röglin, and C. Witt. Theoretical analysis of two aco approaches for the traveling salesman problem. *Swarm Intelligence*, 6(1):1–21, 2012.

[8] S. Kratsch and F. Neumann. Fixed-parameter evolutionary algorithms and the vertex cover problem. *Algorithmica*, 65(4):754–771, 2013.

[9] F. Neumann, D. Sudholt, and C. Witt. Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. *Swarm Intelligence*, 3(1):35–68, 2009.

[10] F. Neumann and C. Witt. Runtime analysis of a simple ant colony optimization algorithm. *Algorithmica*, 54(2):243–255, 2009.

[11] F. Neumann and C. Witt. Ant colony optimization and the minimum spanning tree problem. *Theoretical Computer Science*, 411(25):2406–2413, 2010.

[12] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization:Algorithms and Their Computational Complexity*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[13] L. V. Quintas and F. Supnick. On some properties of shortest Hamiltonian circuits. *The American Mathematical Monthly*, 72(9):977–980, 1965.

[14] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[15] J. Scharnow, K. Tinnefeld, and I. Wegener. The analysis of evolutionary algorithms on sorting and shortest paths problems. *J. Math. Model. Algorithms*, 3(4):349–366, 2004.

[16] T. Stützle. ACO-TSP software package, 2012.

[17] T. Stützle and H. H. Hoos. MAX-MIN Ant system. *Future Generation Computer Systems*, 16(9):889–914, June 2000.

[18] D. Sudholt and C. Thyssen. Running time analysis of ant colony optimization for shortest path problems. *J. Discrete Algorithms*, 10:165–180, 2012.

[19] D. Sudholt and C. Thyssen. A simple ant colony optimizer for stochastic shortest path problems. *Algorithmica*, 64(4):643–672, 2012.

[20] A. M. Sutton and F. Neumann. A parameterized runtime analysis of evolutionary algorithms for the euclidean traveling salesperson problem. In J. Hoffmann and B. Selman, editors, *AAAI*. AAAI Press, 2012.

[21] A. M. Sutton and F. Neumann. A parameterized runtime analysis of simple evolutionary algorithms for makespan scheduling. In *Proceedings of the Twelfth Conference on Parallel Problem Solving from Nature (PPSN 2012)*, pages 52–61. Springer, 2012.

[22] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.