**Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)**

Faster Evolutionary Algorithms by Superior Graph Representation

Benjamin Doerr
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany

Christian Klein
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany

Tobias Storch
Department of Computer Science 2
University of Dortmund
Otto-Hahn-Str. 14
44221 Dortmund, Germany

***Abstract*— We present a new representation for individuals in problems that have cyclic permutations as solutions. To demonstrate its usefulness, we analyze a simple randomized local search and a (1+1) evolutionary algorithm for the Eulerian cycle problem utilizing this representation. Both have an expected run-time of $\Theta(m^2 \log(m))$, where m denotes the number of edges of the input graph. This clearly beats previous solutions, which all have an expected optimization time of $\Theta(m^3)$ or worse (PPSN '06, CEC '04). We are optimistic that our representation also allows superior solutions for other cyclic permutation problems. For NP-complete ones like the TSP, however, other means than theoretical run-time analyses are necessary.**

I. Introduction

Randomized search heuristics not only have many applications but also a high variety. The probably best-known heuristics belonging to this broad class of optimizers are randomized local search (RLS) and evolutionary algorithms (EAs) (cf. [1]). For a wide range of applications a remarkable experimental success of such heuristics (and hybrid algorithms) is reported (see, for example, [8] for the traveling salesman problem). From a theoretical point of view, however, only little is known about randomized search heuristics. The first rigorous probabilistic runtime analyses of EAs investigated the behavior of simple variants thereof on simple classes of functions (see [4]). Even less is known for the behavior of randomized search heuristics in combinatorial optimization. Only recently, analyses of simple EAs on combinatorial optimization problems were presented (see, for example, [13] for the partition problem and [11] for the minimum spanning tree problem). Most of these EAs are intended to optimize pseudo-Boolean objective functions $f : \{0,1\}^n \to \mathbb{R}$ or functions $f : \mathcal{S}_n \to \mathbb{R}$ on the set of all permutations. Nonetheless, it is well-known in evolutionary computation that the appropriate choice of the objective function describing the given problem – and thereby, in particular the choice of the search space – is of huge importance (see [12] for the minimum spanning tree problem).

Many combinatorial optimization problems deal with graphs and subclasses thereof, such as trees. We develop an edge-based representation for graphs which is quite simple and natural and efficient to handle. Moreover, an appropriate mutation operator for our representation is given. We examine how simple randomized search heuristics for the Eulerian cycle problem benefit from our representation to illustrate its advantages.

A. The Eulerian Cycle Problem

The Eulerian cycle problem is a prototypical problem on graphs. It is the generalization of the well-known problem of the Seven Bridges of Königsberg presented by Leonard Euler in 1736, one of the first considerations of graph theory (see [6]). Furthermore, the Eulerian cycle problem is not only an interesting theoretical problem, it is also a subroutine for many established algorithms such as Christofides approximation algorithm for the metric traveling salesman problem [2]. Given an undirected connected graph $G = (V, E)$ a solution to the Eulerian cycle problem is a tour that uses every edge exactly once. Euler already proved that such a tour exists if and only if the degree of each vertex is even. Graphs containing an Eulerian cycle are called Eulerian (graphs). Optimal deterministic algorithms for the Eulerian cycle problem compute such a tour in time $O(|V| + |E|)$ (see [7]).

Since randomized search heuristics are not problem-specific, it is doubted that they can outperform algorithms specifically tailored for a problem. They are, however, typically easy to implement and thus quite popular for real-world applications. Randomized search heuristics for a specific problem may also be able to solve generalizations of the problem. They can also be easily applied to a wide range of not so well understood problems. Investigating them can help to increase the knowledge about such problems or their generalizations and may thus even lead to better problem-specific algorithms (see [5] for generalizations of the Eulerian cycle problem).

B. Previous Work

Neumann [10] investigated the Eulerian cycle problem for the search space of all permutations of the edges. He considers two mutation operators; the "exchange"-operator, which swaps two elements of the permutation and the "jump"-operator, which inserts an element of the permutation at a new position. His objective function counts how many of the leading elements of the permutation still describe a path in the graph. For the exchange-operator, he proves that the RLS and the $(1+1)$-EA have an infinite resp. exponential expected optimization time. Using the jump-operator, both RLS and the $(1 + 1)$-EA have a polynomial expected runtime of $O(|E|^5)$. Doerr, Hebbinghaus, and Neumann [3] analyzed a restricted version of the jump operator. Their modification improves the expected

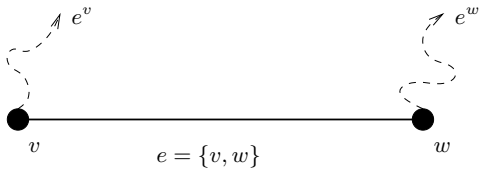

Fig. 1. The edge $e = \{v, w\}$ and its two associated pointers.

optimization time to $O(|E|^3)$ for both RLS and the $(1 + 1)$-EA. They also give a class of graphs for which both heuristics have optimization time $\Omega(|E|^3)$ with high probability.

C. Our Contribution

In this paper, we improve the optimization time to $O(|E|^2 \ln |E|)$ for the RLS and the $(1 + 1)$-EA. To achieve this significant improvement, we do not tweak the mutation operator as in the above mentioned papers, but design a better graph representation for the heuristics to use. We also show that our analysis is tight by giving graphs that exhibit an expected optimization time of $\Omega(|E|^2 \ln |E|)$.

In order to concentrate on the effects of our newly developed graph representation, we only consider two simple randomized search heuristics; the RLS, which just performs local changes, and the so-called $(1 + 1)$-EA, which also performs global changes. This avoids unnecessary complications in the analysis due to the effects of other components of randomized search heuristics. For some optimization problems, a representation dual to the one presented in this paper, i.e., with the roles of vertices and edges exchanged, may be more suitable.

In Section II we introduce our new graph representation for evolutionary computation together with an appropriate mutation operator. We also give a natural and easy evaluable objective function for the Eulerian cycle problem. In Section III we define and analyze a RLS for this representation. This analysis is extended to the $(1+1)$-EA in Section IV. We finish with a summary and some conclusions in Section V.

II. Representation

In this section we propose a new representation of individuals that in particular allows faster computation of an Eulerian cycle.

A. The Search Space

Given an undirected graph $G = (V, E)$ let $v_1, \ldots, v_n$ be the n vertices and $e_1, \ldots, e_m$ be the m edges in an arbitrary but fixed order.

In previous works individuals are represented as permutation of the edges. This permutation-based representation, however, leads to various problems, for example if a graph consists of multiple cycles (cf. [3], [10]).

A representation of paths used in classical graph theory is to name the edges in the order traversed and for each pair of edges name the vertex they share. Based on this we give an edge-based representation that represents a graph as sequence of paths. For each edge, we want to store its two neighboring edges in a path. In other words, for an edge $e = \{v, w\} \in E, v, w \in V$ we need to store its neighboring edge incident to v and its neighboring edge incident to w.

Technically, an individual is represented by an array of $2m$ pointers, two of them associated with each edge, one for each vertex incident to the edge. More precisely, let $e_i = \{v_j, v_k\}, i \in [1..m], j, k \in [1..n], j < k \leq n$. We associate the $(2i - 1)$st and the $2i$th pointer with the edge e_i. The first pointer, denoted $e_i^{v_j}$, is associated with vertex v_j of edge e_i, and the second pointer, denoted $e_i^{v_k}$, is associated with vertex v_k of edge e_i. Figure 1 shows the situation for one edge $e = \{v, w\}$.

A pointer e^w of an edge $e \in E$ should always point to an edge $f \in E$ for which $w \in f$ holds. Otherwise the pointer points to a special value $\perp$. In the first case we write $e^w \to f$, in the second case we write $e^w \to \perp$.

For two edges $e, f \in E$ we write $e \overset{=}{\leftrightarrow} f$, if

$$\exists v \in V : (v \in e) \wedge (v \in f) \wedge (e^v \to f) \wedge (f^v \to e).$$

In other words, we write $e \overset{=}{\leftrightarrow} f$ if the edges e and f have a common vertex v and each of them has a pointer to the other edge.

The initial individual has all pointers pointing to $\perp$. Hence the first individual is just a collection of m connected components (the edges). An individual representing an Eulerian cycle on the other hand has just one connected component of size m. The set of all individuals (of size m) is called the *search space* and denoted by $\mathcal{I}_m$.

B. The Mutation Operator

Having characterized the search space, we now give a canonical mutation operator for it. We denote this mutation operator on $\mathcal{I}_m$ by "$\oplus$". The operator will connect two random edges by adjusting their pointers accordingly (see Figure 2). A formal definition follows.

Definition 1: Let $e, f \in E$ and $e', f' \in E \cup \{\perp\}$ such that $e^v \to e'$ and $f^w \to f'$. If both $e', f' \neq \perp$ and $v = w$ then $e^v \oplus f^w$ changes the pointers of those four edges as follows:

$$e^v := f \quad ; \quad f^w := e$$
$$e'^v := f' \quad ; \quad f'^w := e'$$

If $v = w$ but $e', f' = \perp$, then only $e^v := f; f^w := e$ will be changed, as $\perp$ has no pointers. If $v \neq w$, then the $\oplus$-operator will set all pointers to $\perp$.

From the definition of the $\oplus$-operator and the fact that the initial individual has only $\perp$-pointers, the following Lemma easily follows.

Lemma 2: Assume that an arbitrary number of $\oplus$-operations is applied to the individual with all pointers set to $\perp$. Then the following holds for any edge $e \in E$ containing a vertex $v \in e$.

1) If $e^v \to \perp$ then there exists an edge $f \in E$ with $v \in f$ and $f^v \to \perp$.
2) If $e^v \to f$ and $f \neq \perp$ then $f^v \to e$.

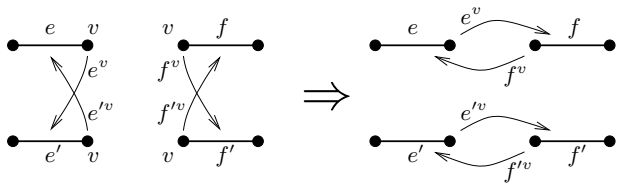

Fig. 2. The $\oplus$-Operator $e^v \oplus f^v$ on four edges incident to a common vertex v. The pointers are depicted as arrows.

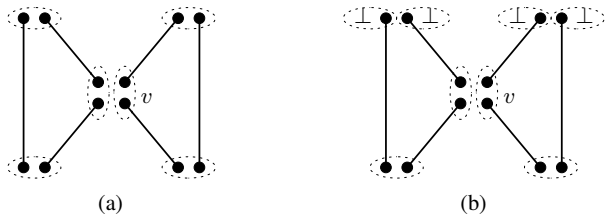

Fig. 3. Two cycles meeting in a common vertex v may pose a problem for simple fitness functions. In case (a) the fitness function f_{ptr} will fail, in case (b) f_{comp} cannot increase. Dashed ellipses mark connected edge-pointers.

Proof: This follow from the fact that if the $\oplus$-operator changes a pointer $e^v := f$ it also changes $f^v := e$. ■

By this Lemma it immediately follows that if $e^v \to f$ then $e \overset{=}{\leftrightarrow} f$ holds.

C. A Fitness Function for the Eulerian Cycle Problem

We have designed the search space and the mutation operator without using any knowledge about the specific problem at hand. To allow randomized search heuristics to solve a given problem, we need to find a good objective function $f : \mathcal{I}_m \to \mathbb{R}$ that assigns to each element of the search space a value. This value should somehow reflect how close an element of the search space is to being a solution to our problem. Such a function is called *fitness function*, since it tells us how fit a certain element of $\mathcal{I}_m$ is. We now give a fitness function for the Eulerian cycle problem whose value will be small for elements close to a solution. The goal of a randomized search heuristic is then to to minimize this function.

Two possible functions seem to be candidates for the fitness function. One is the number of pointers to $\perp$, denoted f_{ptr}, as an Eulerian cycle has no $\perp$-pointers. The other is the number of connected components in the individual, denoted f_{comp}, as an Eulerian cycle is just one connected component.

Unfortunately, they both don't work as expected. To see this, consider a graph consisting of multiple cycles. For f_{ptr}, an individual where each cycle forms its own component will have the same fitness as an Eulerian cycle (see Figure 3(a) for an example). A search heuristic using f_{comp} may also fail, although an individual consisting of just one component is an Eulerian tour and vice versa. To see this, again consider a graph consisting of multiple cycles and an individual where each cycle forms its own component. If each component has exactly one pair of $\perp$-pointers, then, no matter what operation is done, the number of components will stay the same. This is illustrated in Figure 3(b).

Although the above examples look quite similar, the fitness functions fail for different reasons. Indeed, the fitness function we will use is simply the sum of both functions, i.e. $f := f_{ptr} + f_{comp}$. For this fitness function we will prove that both a variant of RLS and a simple EA will successfully generate an Eulerian cycle.

Since an Eulerian tour has no pointers to $\perp$ and consists of just one component, f will assign a fitness of 1 to it. The initial individual has fitness $3m$, as each of the m edges is its own component and has two pointers to $\perp$. If a graph is not Eulerian, it has a fitness of more than 1. Hence we will only consider Eulerian graphs as input.

Note that if we consider multi-graphs, the above given representation and fitness function still work.

RANDOMIZED LOCAL SEARCH
1 Initialize pointers of individual I to $\perp$.
2 **repeat**
3 $I' \leftarrow I$
4 Choose pointers e^v, f^w u. a. r.
5 Apply $e^v \oplus f^w$ to I'.
6 **if** $f(I') \leq f(I)$
7 **then** $I \leftarrow I'$
8 **until** false

Fig. 4. Randomized Local Search.

III. RANDOMIZED LOCAL SEARCH

Randomized local search is one of the simplest randomized search heuristics. It considers a population of size one and produces a single new individual in each generation. To generate the new individual, a single mutation is applied to the current individual. If the fitness of the newly generated individual is not worse than that of the old individual, the new individual replaces the old one. As we are minimizing, this happens if the value assigned to the old individual by f is not lower than the value assigned to the new individual. The mutation and selection is then repeated forever. In the mutation step, two array elements $i, j \in [1..2m]$, corresponding to two pointers e^v, f^w, are chosen uniformly at random. The mutation applied to get the new individual is $e^v \oplus f^w$. Figure 4 shows the pseudo-code for the variant of RLS considered by us. In practice, a stopping criterion is needed, as we cannot run the algorithm forever. Hence, we are interested in the expected number of fitness evaluations until the individual I represents an Eulerian cycle.

A. An Upper Bound on the Optimization Time of RLS

We first analyze the probability that a random mutation will improve the fitness of an individual.

Lemma 3: The probability for a mutation to improve the fitness of an individual with fitness $k > 1$ is $\Omega(\frac{k}{m^2})$.

Proof: First observe that each component in the representation of an individual contributes at least 1 to its fitness. On the other hand, each component can contribute at most 3 (1 for it being a component and 2 for two $\perp$-pointers if it is not a cycle but a path) to the fitness. Hence an individual of fitness k has $\Theta(k)$ components.

We now show that for each component there is at least one fitness-increasing mutation $e^v \oplus f^v$, where e is an edge of the component. This and the fact that there are $\Theta(k)$ components proves the Lemma.

Case 1: The component is a cycle.
Since $k > 1$, this cannot be the only component of the individual. Hence there exists a vertex v which is incident with both an edge of the component and an edge not belonging to the component. Let e and e' be two edges of the component containing the vertex v for which $e \overline{\overline{\leftrightarrow}} e'$ holds. Since each vertex has even degree, there are at least two edges f, f' not belonging to the component and adjacent to v such that either one of them, say f, is the first edge of a component, or they both belong to the same component and $f \overline{\overline{\leftrightarrow}} f'$. In both cases the mutation $e^v \oplus f^v$ will merge the two components e and f belong to. Hence, the number of components decreases and thus the fitness of the individual improves under this mutation.

Case 2: The component is not a cycle.
Then there must be an edge e with $e \overline{\overline{\leftrightarrow}} \perp$ in the component. Let $v \in e$ be the vertex of e for which $e^v \to \perp$. By Lemma 2, there exists another edge f with $v \in f$ and $f^v \to \perp$. But then $e^v \oplus f^v$ will decrease the number of $\perp$-pointers and thus improve the fitness. Observe that either e and f can be edges of the same components, in which case the mutation will change the component to a cycle, or f belongs to another component, in which case the component of e is enlarged. ■

Since RLS will not accept mutations that decrease the fitness of an individual, Lemma 3 tells us how long we need to wait in expectation until a fitness-increasing mutation happens. Because the fitness of the initial individual is known to be $3m$, this allows us to give an upper bound on the expected optimization time.

Theorem 4: Randomized Local Search will produce an Eulerian cycle after an expected number of at most $O(m^2 \log m)$ steps.

Proof: According to Lemma 3 it takes an expected number of

$$O\left(\sum_{k=1}^{m} \frac{m^2}{k}\right) = O(m^2 \log m)$$

steps until the individual consist of only one connected component. If this component forms an Eulerian cycle, we are done. If not, we can apply case 2 from the proof of Lemma 3 to get an Eulerian cycle after an expected number of m^2 additional steps. ■

B. A Lower Bound on the Optimization Time of RLS

To prove that our analysis is tight, let C_m be the graph consisting of a single cycle of $m = n$ edges. For this graph the expected optimization time can be calculated exactly. Define the mth harmonic number as $\mathcal{H}_m := \sum_{i=1}^{m} \frac{1}{i}$.

Theorem 5: The expected optimization time of Randomized Local Search on C_m equals $2m^2\mathcal{H}_m \in \Theta(m^2 \log m)$.

Proof: The first individual consists of the m single edges and all pointers set to $\perp$ and thus has a fitness of $3m$. As long as the individual does not represent an Eulerian cycle, its fitness is a multiple of 3, namely 3 times the number of components, as it cannot contain any sub-cycles.

As long as the fitness is $3m$, there exists exactly one pointer f^v for any fixed pointer e^v, such that $e^v \oplus f^v$ improves the fitness of the individual. Hence with probability $\frac{1}{2m}$ the fitness will improve by 3.

Now assume that the current individual consists of $1 < k < m$ connected components, hence having fitness $3k$. Then there are $2k$ pointers (namely the first and last of each individual) which can be used to improve the fitness. One of those is picked with probability $\frac{2k}{2m}$. If this happens, then the probability that the second pointer is such that the fitness improves is $\frac{1}{2m}$, since there is only one such pointer. Hence, with probability $\frac{2k}{4m^2}$ the fitness improves by 3 in a single mutation. We conclude that the expected time for this improvement is $\frac{2m^2}{k}$.

Thus, to reach fitness 3, we need an expected number of $\sum_{k=2}^{m} \frac{2m^2}{k}$ steps. For the last step, there are only two unused pointers left. The probability to pick one of them as first pointer is $\frac{1}{m}$, and the probability to pick the other unused pointer as second pointer is $\frac{1}{2m}$. Hence we need another $2m^2$ steps in expectation to complete the cycle. In summary, RLS needs an expected number of

$$2m^2 \sum_{k=1}^{m} \frac{1}{k} = 2m^2\mathcal{H}_m \in \Theta(m^2 \log m)$$

steps. ■

$(1+1)$ EVOLUTIONARY ALGORITHM

```
1  Initialize pointers of individual I to ⊥.
2  repeat
3        s ← Pois(λ = 1).
4        I' ← I.
5        for i ← 1 to s + 1
6             do
7                  Choose pointers e^v, f^w u. a. r.
8                  Apply e^v ⊕ f^w to I'.
9        if f(I') ≤ f(I)
10         then I ← I'
11   until false
```

Fig. 5. The $(1+1)$ Evolutionary Algorithm.

IV. THE (1+1)-EA

The perhaps simplest evolutionary algorithm is the $(1+1)$-EA. In contrast to RLS, it allows for more than one $\oplus$-operation in each mutation step. The "classical" $(1+1)$-EA

on bit-strings of length n independently flips each bit with probability $1/n$. When using more complex representations, for example permutations [10], this behavior must be simulated. To do this, a number s is chosen at random according to a Poisson distribution[1] $\mathrm{Pois}(\lambda = 1)$ with parameter $\lambda = 1$. Then the mutation step of RLS, namely choosing two pointers uniformly at random and applying the $\oplus$-operation, is done $s+1$ times instead of just once. The reason why $s+1$ instead of s mutations are performed is to prevent steps in which nothing happens. Figure 5 shows the pseudo-code of the $(1+1)$-EA.

A. An Upper Bound on the Optimization Time of the $(1+1)$-EA

We now show that the $(1+1)$-EA is at least as fast as RLS. To do this, we simply analyze the time it takes to "simulate" RLS, i.e., we ignore all mutations that do more than one $\oplus$-operation.

Theorem 6: The $(1+1)$-EA will produce an Eulerian cycle after an expected number of at most $O(m^2 \log m)$ steps.

Proof: A mutation applying the $\oplus$-operator more than once will only be accepted if it does not worsen the fitness of the individual. Since we are looking for an upper bound on the expected optimization time, we can thus safely ignore all mutations that apply more than one $\oplus$-operation to the individual.

If $s = 0$, then the $\oplus$-operator will be applied exactly once. This occurs with a constant probability, namely $e^{-1}\frac{1^0}{0!} = e^{-1}$ by definition of the Poisson distribution (cf. [9]). Hence, in expectation each eth step exactly one mutation is performed.

With this the proof of Theorem 4 can be applied. It follows that the expected optimization time is at most $e \cdot O(m^2 \log m) = O(m^2 \log m)$. ∎

B. A Lower Bound on the Optimization Time of the $(1+1)$-EA

While Theorem 6 shows that the expected optimization time of the $(1+1)$-EA is not worse than that of RLS, we will now show that on the graph C_m it indeed needs an expected number of $\Omega(m^2 \log m)$ steps.

Theorem 7: The expected optimization time of the $(1+1)$-EA on C_m is $\Theta(m^2 \log m)$.

Proof: This proof loosely follows a proof of Droste, Jansen and Wegener [4]. They show that the $(1+1)$-EA needs an expected optimization time of $\Omega(n \log n)$ to optimize any linear function with non-negative weights on $\{0,1\}^n$. Some arguments are similar to the coupon collector's problem (cf. Motwani and Raghavan [9]).

Let T denote the random variable describing the number of steps needed by the $(1+1)$-EA. The expected value of T can then be bounded by

$$\mathbb{E}(T) = \sum_{i=1}^{\infty} iP(T = i) \geq tP(T \geq t)$$

for a fixed $t \geq 1$. The probability $P(T \geq t)$ that the $(1+1)$-EA needs at least t steps, however is the same as the probability that the $(1+1)$-EA will not finish after $t-1$ steps. Hence if we can bound this probability by some value p from below for a given t, we get a lower bound on the expected optimization time.

To get this bound, first consider a vertex v of C_m and the two edges e, f containing v. Then the pointers e^v and f^v have to be chosen at least once for a particular $\oplus$-operation to construct an Eulerian cycle. The probability that exactly this mutation happens is $\frac{2}{2m} \cdot \frac{1}{2m} = \frac{1}{2m^2}$.

The probability that during a single step exactly $s+1$ $\oplus$-operations are performed is $\frac{1^s}{es!}$ by the definition of the Poisson distribution (cf. [9]). Since each of the $s+1$ $\oplus$-operations are done independently, the probability that a fixed $\oplus$-operation is performed in a certain step is at most $\frac{s+1}{2m^2}$. Hence the probability that a fixed $\oplus$-operation is performed in one mutation is at most

$$\sum_{s=0}^{\infty} \frac{1}{e \cdot s!} \cdot \frac{s+1}{2m^2} \leq \frac{1}{m^2}$$

since $\sum_{s=0}^{\infty} \frac{s+1}{s!} = 2e$.

The probability that a fixed combination was not chosen at all during $t-1$ steps is then at least $(1 - \frac{1}{m^2})^{t-1}$. But then the probability that the combination is chosen at least once in the $t-1$ steps considered is at most $1 - (1 - \frac{1}{m^2})^{t-1}$. Since we have m different combinations, the probability that all combinations are considered is at most $(1-(1-\frac{1}{m^2})^{t-1})^m$. Hence, the probability that at least one of the combinations is never chosen during the $t-1$ steps considered is at least $1-(1-(1-\frac{1}{m^2})^{t-1})^m$. We now choose $t := 1+(m^2-1)\log m$ to obtain

$$1 - \left(1 - \left(1 - \frac{1}{m^2}\right)^{(m^2-1)\log m}\right)^m \geq 1 - e^{-1}.$$

Hence, for this value of t we have $p \geq 1 - e^{-1}$.

Thus the expected number of steps needed is bounded from below by

$$(1 - e^{-1}) \cdot (1 + (m^2 - 1)\log m) = \Omega(m^2 \log m).$$

∎

V. Conclusions

We have given a new edge-based representation for graphs together with a canonical mutation operator. To show the benefits of this representation we have rigorously analyzed a variant of RLS and a simple EA for the Eulerian cycle problem. Our representation yields much faster randomized search heuristics than previously studied representations for this problem (cf. [3], [10]). We expect this type of graph representation to be useful for other problems as well.

References

[1] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[2] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," in *Algorithms and complexity : New directions and recent results*, J. Traub, Ed. Academic Press, Inc., 1976, p. 441.

[1] The Poisson distribution with $\lambda = 1$ is used because it is the limit of the Binomial distribution for n trials with probability $\frac{1}{n}$ each.

[3] B. Doerr, N. Hebbinghaus, and F. Neumann, "Speeding up evolutionary algorithms through restricted mutation operators," in *Proceedings of the 9th International Conference on Parallel Problem Solving From Nature (PPSN)*, ser. Lecture Notes in Computer Science, vol. 4193. Springer, 2006, pp. 978–987.

[4] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theoretical Computer Science*, vol. 276, pp. 51–81, 2002.

[5] J. Edmonds and E. L. Johnson, "Matching, euler tours and the chinese postman," *Mathematical Programming*, vol. 5, pp. 88–124, 1973.

[6] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii academiae scientiarum Petropolitanae*, vol. 8, pp. 128–140, 1741.

[7] C. Hierholzer, "Ueber die Möglichkeit, einen Linenzug ohne Wiederholung und ohne Unterbrechung zu umfahren," *Mathematische Annalen*, vol. 6, pp. 30–32, 1873.

[8] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial Intelligence Review*, vol. 13, pp. 129–170, 1999.

[9] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[10] F. Neumann, "Expected runtimes of evolutionary algorithms for the Eulerian cycle problem," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press, 2004, pp. 904–910.

[11] F. Neumann and I. Wegener, "Randomized local search, evolutionary algorithms, and the minimum spanning tree problem," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, ser. Lecture Notes in Computer Science, vol. 3102. Springer, 2004, pp. 713–724.

[12] G. R. Raidl and B. A. Julstrom, "Edge sets: an effective evolutionary coding of spanning trees." *IEEE Trans. Evolutionary Computation*, vol. 7, pp. 225–239, 2003.

[13] C. Witt, "Worst-case and average-case approximations by simple randomized search heuristics," in *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, ser. Lecture Notes in Computer Science, vol. 3404. Springer, 2005, pp. 44–56.