Evolutionary Placement of Continuously Operating Reference Stations of Network Real-Time Kinematic

Maolin Tang
School of Electrical Engineering and Computer Science
Queensland University of Technology
2 George Street, Brisbane, QLD 4001, Australia
m.tang@qut.edu.au

***Abstract*—Network RTK (Real-Time Kinematic) is a technology that is based on GPS (Global Positioning System) or more generally on GNSS (Global Navigation Satellite System) observations to achieve centimeter-level accuracy positioning in real time. It is enabled by a network of Continuously Operating Reference Stations (CORS). CORS placement is an important problem in the design of network RTK as it directly affects not only the installation and running costs of the network RTK, but also the Quality of Service (QoS) provided by the network RTK. In our preliminary research on the CORS placement, we proposed a polynomial heuristic algorithm for a so-called *location-based CORS placement problem*. From a computational point of view, the location-based CORS placement is a large-scale combinatorial optimization problem. Thus, although the heuristic algorithm is efficient in computation time it may not be able to find an optimal or near optimal solution. Aiming at improving the quality of solutions, this paper proposes a repairing genetic algorithm (RGA) for the location-based CORS placement problem. The RGA has been implemented and compared to the heuristic algorithm by experiments. Experimental results have shown that the RGA produces better quality of solutions than the heuristic algorithm.**

***Keywords*—genetic algorithm, reference station placement, network RTK**

I. Introduction

Network RTK (Real-Time Kinematic) is a technology that is based on GPS (Global Positioning System) or more generally on GNSS (Global Navigation Satellite System) observations to achieve centimeter-level accuracy positioning in real time [1], [2], [3]. It is underpinned by a network of Continuously Operating Reference Stations (CORS), which continuously stream their satellite observations to a central server where a network RTK software is used to fix the ambiguities of the satellites and then generates corrections from those satellite observations. The users connect to the network RTK server to get the correction data and use it computes their positions. In this way, the users can achieve centimeter accuracy positioning services.

The placement of the CORS is a very important issue in the design of network RTK as it will directly affect the QoS and cost of the network RTK. In order to make sure that the network RTK can provide accurate correction data for all the users, it must be guaranteed that there will be sufficient CORS such that for each and every user there will be at least at least one CORS which is geographically close to the user. However, the establishment and maintenance of CORS is expensive, typically tens of thousands of Australian dollars each with annual operational cost of approximately 10% of the CORS. Therefore, it is desirable to minimize the total number of CORS without compromising the quality of the real-time positioning services.

There are two categories of CORS placement problems in network RTK. One is a so-called *location-oriented CORS placement*. Given the CORS candidate sites and the distribution of users, the location-oriented CORS placement problem is to determine at which CORS candidate site we need to place a CORS such as that for each and every user there will be at least CORS such that the Euler distance between the user and the CORS site is not greater than D_{max} and the total number of CORS is minimal. Another CORS placement problem is so called *area-oriented CORS placement*. Given an area of any shape where the network RTK needs to cover, the area-oriented placement problem is to find a placement of CORS in the area such that at any position in the area there is at least one CORS such that the distance between them is not greater than D_{max} and the total number of CORS is minimal. This paper focuses on the location-based CORS placement problem.

In our preliminary research on the location-oriented CORS placement, we have proposed a heuristic algorithm for the problem. From a computational point of view, the location-based CORS placement is a large-scale combinatorial optimization problem. Thus, although the heuristic algorithm is efficient in computation time it may not be able to find an optimal or near optimal solution. Aiming at improving the quality of solutions, this paper proposes a repairing genetic algorithm (RGA) for the location-based CORS placement problem. The RGA has been implemented and compared to the heuristic algorithm by experiments. Experimental results have shown that the RGA constantly produces better quality of solutions than the heuristic algorithm.

The paper is organized as follows. Firstly, we formulate the location-oriented CORS placement problem in Section II.

Then, we discuss related work in Section III before we present the RGA in Section IV. Finally, we conclude the research and talk about future work on the location-oriented CORS placement problem.

II. Problem Formulation

In the location-oriented CORS placement problem, the locations of all the users and all the CORS candidates are given. In order to guarantee that all the users will get centimeter accuracy positioning services, we must make sure that there will be at least one CORS candidate is selected such that the distance between the user and the CORS candidate is not greater than a parameter D_{max}.

Figure 1 illustrates the location-oriented CORS placement problem. In the figure, u_1, u_2, $\cdots$, u_9 represent users, and c_1, c_2, $\cdots$, u_5 are CORS candidates. The radius of those big broken-line circles is D_{max}. Thus, those big broken-line circles show the coverage of the CORS candidates.

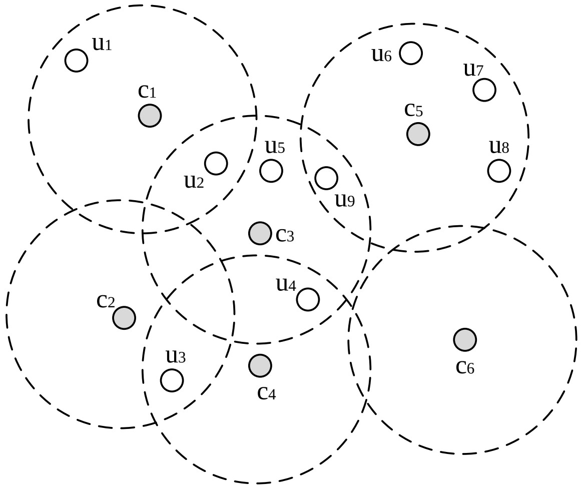

Fig. 1. Illustration of the location-oriented CORS placement problem

To model the location-oriented CORS placement problem as a graph-theoretic problem, we define some symbols that will be used in the modeling in Table I.

TABLE I
Symbols used in the problem formulation

U	the set of users
C	the entire set of CORS candidates
D_{max}	the maximal spacing constraint
$u = (x_u, y_u) \in U$	the location of user u
$c = (x_c, y_c) \in C$	the location of CORS candidate c
C^*	a subset of C where a CORS will be placed
$\|C^*\|$	the number of CORS candidates in C^*

The location-oriented CORS placement problem can be represented by a bipartite graph [4] $G = (V_1 \bigcup V_2, E)$, where $V_1 = U$, $V_2 = C$. An edge $(u, v) \in E$, if and only if $\exists u = (x_u, y_u) \in U$, $\exists c = (x_c, y_c) \in C$, and $\sqrt{(x_u - x_c)^2 + (y_u - y_c)^2} \leq D_{max}$. This bipartite graph is called *coverage graph*. Figure 2 is the coverage graph of the location-oriented CORS placement problem in Figure 1.

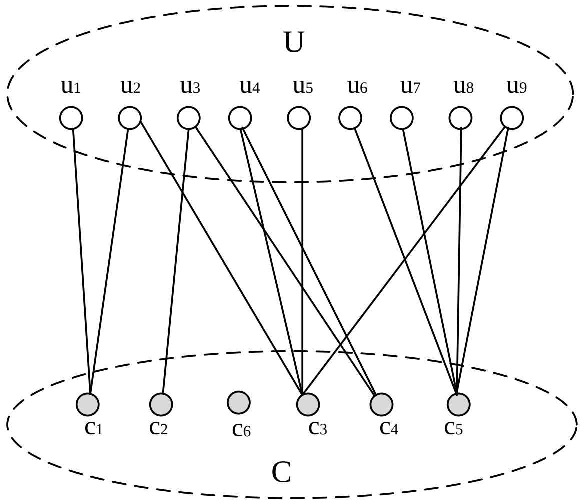

Fig. 2. Illustration of coverage graph

It is assumed that for any user there exists at least one CORS candidate such that the distance between the user and the candidate CORS does not exceed D_{max}. The location-based CORS placement problem can be transformed into the following graph-theoretic problem:

Given a coverage graph $G = (C \bigcup U, E)$, find $C^* \subseteq C$, such that $|C^*|$ is minimum and $\forall u \in U\ \exists v \in C^*$ such that $(u, v) \in E$.

Theorem 2.1: The location-based CORS placement problem is NP-complete.

Proof: In order to prove the location-based CORS placement problem (CP) is NP-complete, we show the location-based CORS placement problem (LP) is equivalent to the Set Cover Problem (SCP) [5] under polynomial reductions.

From CP to SCP:

Given an instance of CP, $G = (C \bigcup U, E)$, construct a SCP $(\overline{S}, \overline{U})$ as follows: the universe $\overline{U} = U$ and the family of subsets is $\overline{S} = \{\overline{S_1}, \overline{S_2}, \cdots, \overline{S_n}\}$, where $\overline{S_i} = \{u_j | (c_i \in C) \& (u_j \in U) \& ((c_i, u_j) \in E)\}$ and $1 \leq i \leq n = |C|$.

If C^* is the minimal CORS set of a CP, then $\overline{S}^* = \{\overline{S_1}^*, \overline{S_2}^*, \cdots, \overline{S_{|C^*|}}^*\}$ is the minimal set cover of the corresponding SCP, where $\overline{S_i}^* = \{u_j | (c_i \in C^*) \& (u_j \in U) \& ((c_i, u_j) \in E)\}$.

From SCP to CP:

Let $(\overline{S}, \overline{U})$ be an instance of SCP with the universe $\overline{U}$ and the family of subsets $\overline{S} = \{\overline{S_i} | i \in I\}$. Construct a CP $G = (C \cup U, E)$ as follows: the user set $U = \overline{U}$; the CORS candidate set $C = \overline{S}$; and edge $(\overline{S_i}, u_j) \in E$ if and only if $u_j \in \overline{S_i}$.

If $\overline{S}^* = \{\overline{S_1}^*, \overline{S_2}^*, \cdots, \overline{S_k}^*\}$ is the minimal cover set of a SCP, then the corresponding CORS set of $\overline{S}^*$ is the minimal

CORS for the corresponding CP.

■

III. Related Work

There are various placement problems in the context of communication network planning and deployment. Although the placement problems are different from each other in terms of objectives and constraints, they all transformed into a problem of finding the locations of communication devices, such as base stations [6], [7], [8], [9], [10], reference stations [11], [12], relay stations [13], [14], cache [15], access point [16], [17] and gateways [18], [19], [20]. Placement problems are very important in communication network planning and deployment as they determine the cost of building a communication network and the performance of the communication network.

The placement problems can be categorized into *discrete placement problems* and *continuous placement problems*. In discrete placement problems, candidate locations of the communication devices are given. Thus, discrete placement problems are selecting a set of locations among those candidate locations such that the objectives are optimized subject to some constraints. The search space of the discrete placement problems is discrete. From a computational point of view, discrete placement problems are constrained combinatorial optimization problems. Thus, various combinatorial optimization techniques are applied to solve the discrete placement problems.

In continuous placement problems, in contrast, there are no candidate locations given for the communication devices. Thus, continuous placement problems are find locations on a continuous area such that objectives are optimized subject to some constraints. Existing approaches to the continuous placement problems are to transform the continuous placement problems into a discrete placement problem by modeling a continuous area using a grid graph where possible locations of reference stations are restricted to the gird points of the grid graph. In this way, a reference station placement problem can be transformed into a combinatorial optimization problem.

Since the location-based CORS placement is NP-hard, a heuristic algorithm (HA) was proposed in [11]. The HA starts with a solution in which every CORS candidate has been placed a CORS. Then, it systematically identifies and removes redundant CORS. A CORS is *redundant* if all the users that are covered by the CORS are also covered by at least another CORS. A user is said to be covered by a CORS if the distance between the user and the CORS is less than or equals to D_{max}. The heuristic algorithm terminates when no redundant CORS can be found. Algorithm 1 is the description of the HA:

It can be seen from the algorithm description that the HA is a hill-climbing algorithm. It is fast in computation time. However, this may be trapped at a local optimum during the exploration. Thus, we propose a Repairing Genetic Algorithm (RGA) for the CORS placement problem in the following section.

Algorithm 1: A HA for the CORS placement algorithm

```
Input  : the set of CORS candidate sites $C$ and the set
         of users $U$
Output: a CORS placement that covers all the users in
         $U$, satisfying the maximum distance constraint
         $D_{max}$
1 construct the coverage graph $G = \langle C \bigcup U, E \rangle$;
2 for $i$ = 1 to $|C|$ do
3     if the $i^{th}$ CORS candidate is redundant then
4         remove it from $C$;
5         remove all the edges adjacent to it from $E$;
6     end
7 end
8 output $C$.
```

IV. The RGA

This section presents a RGA for the location-oriented CORS placement problem. The RGA uses a repairing technique to fix up any infeasible solution that may be generated in the initial population generation, and any infeasible solution that may be generated by the genetic operators. A solution is said to be feasible, if in the solution for every user there exists at least CORS such that the distance between the user and the CORS is less than or equals to D_{max}; otherwise, it is infeasible.

A. The encoding scheme

In the RGA we use a binary string $c_1c_2\cdots c_n$ to represent a solution to a location-oriented CORS placement problem, where c_i corresponds to a CORS candidate, and $1 \leq i \leq n = |C|$.

$$c_i = \begin{cases} 1, & \text{if CORS candidate } c_i \text{ is selected;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

B. Fitness function

Since the RGA is a repairing one, at the end of each generation all the individuals are feasible. Thus, when defining the fitness function we only need to consider feasible solutions. In addition, the quality of a CORS placement depends on the number of CORS in the placement solution. The more CORS in the placement, the worse the quality of the solution is. Equation 2 is the fitness function that is used to measure the fitness value of an individual $p = c_1c_2\cdots c_n$.

$$fitness(p) = 1 - \frac{\sum_{i=1}^{n} c_i}{n} \tag{2}$$

where $n = |C|$, representing the total number of CORS candidates, and $\sum_{i=1}^{n} c_i$ gives the total number of CORS in p. The fewer the number of CORS in p, the greater the fitness value is.

Property 4.1: For any feasible CORS placement p, $0 \leq fitness(p) \leq 1$.

C. Initial population generation

The RGA begins by randomly creating a population of individuals, each of which is a solution to the CORS placement problem. When randomly generating an individual, it randomly picks up a value (0 or 1) for each of the gens in the chromosome. It is important to note that an individual generated in this manner may not be a feasible solution. If it is the case, the RGA will use a repairing technique, which will be detailed in the following, to transform it into a feasible one.

D. Repairing technique

1) The representation of the coverage graph: A coverage graph $G = (V, E)$, where $V = U \bigcup C$, can be represented by an $m \times n$ adjacency matrix $A = [a_{ij}]_{m \times n}$, where $m = |U|$, $n = |C|$, and

$$a_{ij} = \begin{cases} 1, & \text{if } u_i \in U,\ c_j \in C, \text{ and } (u_i, c_j) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

For example, the coverage graph shown in Figure 2 can be represented by the following adjacency matrix:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

2) The coverage of an individual: The coverage of an individual $p = c_1 c_2 \cdots c_n$ can be represented by a matrix, $B = [b_{ij}]_{m \times n}$, where $b_{ij} = a_{ij} \times c_j$, $1 \leq i \leq m$, and $1 \leq j \leq n$. From a coverage matrix, we can easily identify if a user is covered in a solution and if a CORS is placed at a CORS candidate in a solution. For example, the coverage of individual $p = 110100$ for the location-oriented CORS placement problem shown in Figure 1 is

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Property 4.2: A user is covered in a solution, there exists at least 1 in the corresponding row of the coverage matrix; otherwise, it is not covered by the solution.

Property 4.3: If a CORS is placed at a CORS candidate, then there exists at least one 1 in the corresponding column in the coverage matrix; otherwise, there is no CORS placed at the CORS candidate in the solution.

3) The repairing technique: The repairing technique is used whenever an infeasible individual is generated. In order to reduce the computation time, the RGA combines the feasibility checking process with the repairing process.

In order to check the feasibility of an individual and repair it, if it is not feasible, firstly we need to build the coverage matrix of the individual. Then, we check the rows in the coverage matrix one by one. If there exists any row in which there is no 1, the individual is not feasible as there is no CORS that can be used as a qualify reference station for the user. To fix up the problem, we randomly selects a 1 in the corresponding row in the bipartite graph, and changes the vale of the corresponding gene in the individual from 0 to 1. Algorithm 2 depicts the repairing process.

Algorithm 2: Repairing Algorithm

```
   Input  : an infeasible solution, $C = c_1 c_2 \cdots c_n$
   Output: a feasible solution, $C' = c'_1 c'_2 \cdots c'_n$
1  $C'$ = $C$;
2  build the coverage matrix of $C'$;
3  for $i$ = 1 to $m$ do
4      if user i is not covered then
5          for $j$ = 1 to $n$ do
6              if user $u_i$ can be covered by CORS candidate
                   $c'_j$ then
7                  $c'_j$ = 1;
8                  update the coverage matrix;
9              end
10         end
11     end
12 end
13 output $C'$.
```

E. Crossover operator

Linkage problem is a potential problem in the RGA as in the encoding scheme each gene represents a CORS candidate and some CORS candidates that are geographically close to each other may interact each other and form some good schemata (building blocks). However, in the encoding scheme those genes may be far apart from each other and therefore those good schemata may be broken if classical one-point or two-point crossover is used.

There are a few ways to deal with linkage problems, and perhaps the simplest way is to use the uniform crossover. The uniform crossover evaluates each gene in the parent chromosomes for exchange with a probability of 0.5. In the RGA we use a modified uniform crossover. Rather than giving the same opportunity to the two parent chromosomes contribute to the child chromosome, we adopt a mixing ratio, which is in proportional to the fitness values of the two parent chromosomes. This is based an assumption that a chromosome that has a greater fitness value contains more useful information. It is expected that the modified crossover results in a more complete search of the design space with maintaining the exchange of good schemata.

Below is the algorithm of the crossover that is used in our RGA:

Algorithm 3: Crossover algorithm

```
Input  : two parent chromosomes, $C^i = c^i_1 c^i_2 \cdots c^i_n$ and
         $C^j = c^j_1 c^j_2 \cdots c^j_n$
Output: one child chromosome, $C^k = c^k_1 c^k_2 \cdots c^k_n$
1  $f^i = fitness(C^i)$;
2  $f^j = fitness(C^j)$;
3  for $q$ = 1 to $n$ do
4      randomly generate a real value between 0 and 1, $r$;
5      if $r < f^i/(f^i + f^j)$ then
6          $c^k_q = c^i_q$;
7      end
8      else
9          $c^k_q = c^j_q$;
10     end
11 end
12 output $C^k$.
```

The child chromosome may not be feasible solution. Thus, if a child chromosome is not a feasible solution, the repairing technique is used to make it feasible.

F. Mutation operator

The mutation operator simply randomly picks up a gene in the chromosome and inverts the value of the chosen gene. Below is the algorithm of the mutation operator:

Algorithm 4: Mutation algorithm

```
Input  : a chromosome, $C = c_1 c_2 \cdots c_n$
Output: a mutated chromosome, $C' = c'_1 c'_2 \cdots c'_n$
1  $C' = C$;
2  randomly generate a number between 1 and $n$, $i$;
3  replace $c'_i$ with $\bar{c'_i}$;
4  output $C'$.
```

The mutated chromosome may no longer be feasible after the mutation. Thus, if this happens, the repairing technique is used to make it feasible.

G. The description of the RGA

The RGA uses a repairing technique to convert any infeasible individual that are generated by the initial population generation procedure and the genetic operators into a feasible one. The selection strategy used in the RGA is the roulette selection. Algorithm 5 is a high-level description of the RGA.

Algorithm 5: The RGA

```
Input  : a set of CORS candidates $C$, a set of users $U$,
         and $D_{max}$
Output: a feasible CORS placement that covers all the
         users in $U$, satisfying the maximum distance
         constraint $D_{max}$
1  generate a population of $PopSize$ individuals, $P$;
2  for each individual in $P$ do
3      if it is not feasible then
4          convert it into a feasible solution using the
           repairing technique;
5      end
6  end
7  initialize the best individual in the past;
8  while the termination condition is not true do
9      for each individual in $P$ do
10         calculate its fitness value;
11     end
12     for each individual in $P$ do
13         use the roulette selection to select another
           individual to pair up;
14     end
15     for each pair of parents do
16         probabilistically use the crossover operator to
           produce an offspring;
17     end
18     for each individual in $P$ do
19         if it is not feasible then
20             convert it into a feasible solution using the
               repairing technique;
21         end
22     end
23     for each individual in $P$ do
24         probabilistically use the mutation operator to
           mutate a randomly chosen gene in the individual;
25         if the mutated individual is not feasible then
26             convert it into a feasible solution using the
               repairing technique;
27         end
28     end
29     find the best individual in $P$;
30     if the best individual in $P$ is better than the best
       individual in the past then
31         replace the best individual in the past with the
           best individual in $P$;
32     end
33 end
34 decode the best individual in the past and output it.
```

V. Evaluation

This section evaluates the performance and scalability of the RGA by conducting an empirical study. In order to conduct the empirical study, we have implemented it in $C\#$ on Microsoft Visual Studio 2010. Since there are no benchmarks available for the location-oriented CORS placement problem, we have developed a problem which can randomly generate test problems. We have also implemented the HA presented in Algorithm 1 in Section III in the same programming language on the same platform as we need to use the solutions generated by the HA as benchmarks to evaluate the quality of solutions produced by the RGA.

A. Test problems generation

In order to evaluate the RGA, we have developed a $C\#$ program that can randomly generate location-oriented CORS placement problems of different characteristics. Given the number of users, the number of CORS candidates, boundaries of the locations of users and CORS candidates, and the value of D_{max}, the program randomly generates a location for each of the users and the CORS candidates.

The program can guarantee that for every test problem generated there exists a feasible solution. This is achieved by the following process. Firstly, the program randomly generates the given number of CORS candidates within the boundaries. Then, it randomly generates a location for a user within the boundaries. Once the location of a user is generated, the program immediately checks if there exists at least one CORS candidate at which if a CORS is deployed the network RTK system can use it as a qualified reference station by checking if the distance between the user and the CORS candidate is not greater than D_{max}. If there is no such a CORS, the program discards the location and re-generates a new location for the user. This is repeated until an appropriate location is found for the user. This process is repeated until all the users' locations are generated.

In the evaluation, all the locations of the users and the CORS candidates are bounded by a 600km $\times$ 400km rectangular and the value of D_{max} is 70km.

B. RGA parameters

In the evaluation, all the RGA parameters were fixed. Table II shows the RGA parameters used in the evaluation.

TABLE II
RGA PARAMETERS

Parameter	Value
population size ($PopSize$)	100
crossover rate (p_c)	0.95
mutation rate (p_m)	0.05
termination condition	no improvement in 15 consecutive generations

C. Scalability of the RGA

The computation time of the RGA depends on both the number of users and the number of CORS candidates. Thus, in order to evaluate the scalability of the RGA we conducted two sets of experiments. In a set of experiments, we gradually increased the number of users without changing the number of CORS candidates to see how the computation time of the RGA would increase. Figure 3 shows how the computation time of the RGA increased when the number of users increased from 100 to 1000 while the number of CORS candidates was fixed to 100. The computation time shown in the figure is the average computation time of 30 runs. It can be seen from the figure that the average computation time of the RGA increased

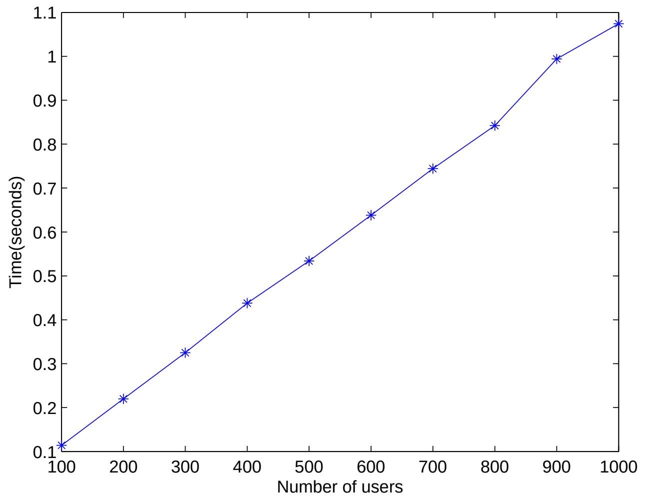

Fig. 3. The computation time of the RGA with the number of CORS candidates

from 0.11 seconds to 1.07 seconds linearly when the number of users increased from 100 to 100.

In the other set of experiments, we gradually increased the number of CORS candidates without changing the number of users to monitor how the computation time would increase. Figure 4 displays the increasing trend of the computation time of the RGA when the number of CORS candidates increased from 100 to 1000 while the number of users was fixed to 100. It can be seen from the figure that the average computation time of the RGA increased from 1.07 seconds to 8.24 seconds linearly when the number of CORS candidates increased from 100 to 1000.

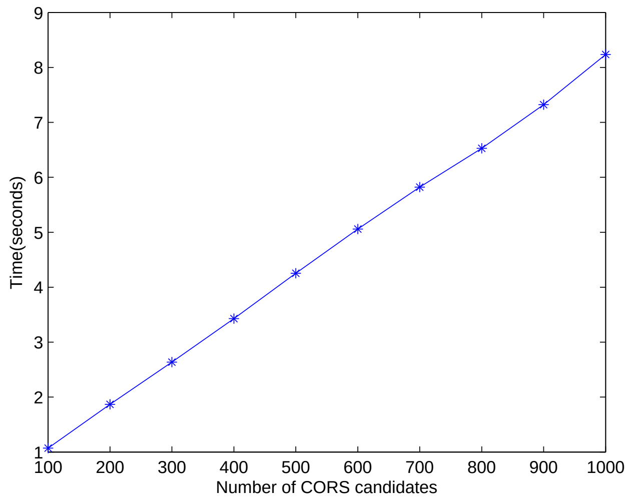

Fig. 4. The computation time of the RGA with the number of users

Based on the above experimental results, it is concluded that the RGA can scale up very well when the size of the location-based CORS placement problem increases.

D. Performance of the RGA

In order to evaluate the performance of the RGA, we randomly generated a number of test problems of various configurations. We used both the RGA and the HA to solve those randomly generated test problems and recorded the solutions obtained by the two algorithms. Considering the stochastic nature of the RGA, we ran the RGA for 30 times for each of the test problems. Tables III and IV show the statistics of the experimental results. Specifically, Table III shows statistics of the experimental results for ten test problems where the number of users ranged between 100 and 1000 and the number of CORS candidates was fixed to 100. Table IV shows statistics of the experimental results for other ten test problems in which the number of CORS candidates ranged between between 100 and 1000 while the number of users was fixed to 100.

TABLE III
COMPARISON OF THE HA AND THE RGA WITH RESPECT TO DIFFERENT NUMBERS OF USERS

Test	U	C	HA	RGA			
Problem	(#)	(#)	Sol	Ave	Best	Worst	StdDev
1	100	100	12	11.97	11	12	0.183
2	200	100	17	13.87	13	14	0.346
3	300	100	14	13.73	13	14	0.450
4	400	100	16	14.70	14	15	0.466
5	500	100	18	13.87	13	15	0.434
6	600	100	17	14.90	14	16	0.481
7	700	100	17	14.97	14	16	0.490
8	800	100	19	14.33	14	15	0.479
9	900	100	15	14.80	13	15	0.484
10	1000	100	16	14.40	14	15	0.498

TABLE IV
COMPARISON OF THE HA AND THE RGA WITH RESPECT TO DIFFERENT NUMBERS OF CORS CANDIDATES

Test	U	C	HA	RGA			
Problem	(#)	(#)	Sol	Ave	Best	Worst	StdDev
1	100	100	16	14.40	14	15	0.498
2	100	200	18	14.90	14	16	0.548
3	100	300	17	14.90	14	16	0.481
4	100	400	18	15.17	14	16	0.461
5	100	500	17	14.90	14	16	0.481
6	100	600	17	14.00	14	16	0.371
7	100	700	17	15.37	14	16	0.556
8	100	800	17	15.00	14	16	0.643
9	100	900	17	15.10	14	16	0.481
10	100	1000	16	15.23	14	16	0.568

It can be observed from the tables that

1) the average solution of the RGA was constantly better than the solution obtained by the HA for all the tested problems.
2) the worst solution of the RGA was still better than, or as good as, the solution produced by the HA. In fact, the RGA generated a worst solution that is still better than the solution generated by the HA for 16 out of the 20 test problems.

In addition, we observed that both the RGA and the HA were always able to generate a feasible solution for all the test problems during the experiments. Thus, we can conclude that the RGA is effective and efficient.

VI. CONCLUSION AND FUTURE WORK

This paper has proposed an RGA for the location-oriented CORS placement problem, and evaluated the performance of the RGA by experiments. The experimental results have shown that the RGA always produced much better results than the HA for all the randomly generated test problems. In addition, this paper has also studied the scalability the RGA. Since the computation time of the RGA depends on both the number of users and the number of CORS candidates, two sets of experiments have been conducted. One experiment was to look at the how the computation time of the RGA would increase when the numbers of users increased without changing the number of CORS candidates. The experimental results have shown that the computation time of the RGA increased linearly when the number of users increased. Another experiment was to study how the computation time of the RGA would increase when the number of CORS candidates increased without changing the number of users. The experimental results have also shown that the computation time of the RGA increased linearly when the number of CORS candidates increased. Thus, it can be concluded that the RGA is scalable.

This is the first attempt to use evolutionary computation to tackle the location-oriented CORS placement problem. In the future we will study how to further improve the RGA using more effective crossover operators. In addition, we will investigate how to hybrid the RGA with the HA to further improve the quality of solutions while reducing the computation time. Finally, the proposed RGA uses a repairing technique to handle infeasible solutions. In the future we will look at other strategies to handle infeasible solutions.

ACKNOWLEDGMENT

The author would like to thank the anonymous reviewers for your valuable comments and suggestions.

This research was carried out as part of the activities of, and funded by, the Cooperative Research Centre for Spatial Information (CRC-SI) through the Australian Government's CRC Programme (Department of Innovation, Industry, Science and Research).

REFERENCES

[1] C. Rizos, "Network rtk research and implementation - a geodetic perspective," *Journal of Global Positioning Systems*, vol. 1, no. 2, pp. 144–150, 2003.
[2] L. Wanninger, "Gps on the web: Virtual reference stations (vrs)," *GPS Solutions*, vol. 7, pp. 143–144, 2003.
[3] C. M. Fotopoulos, G, "An overview of multi-reference station methods for cm-level positioning," *GPS Solutions*, vol. 4, no. 3, pp. 1–10, 2001.
[4] G. Chartrand, *Introductory Graph Theory*. Dover, New York, 1985.
[5] M. Garey and D. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman, 1979.
[6] M. Wright, "Optimization methods for base station placement in wireless applications," in *Vehicular Technology Conference, 1998. VTC 98. 48th IEEE*, vol. 1, May 1998, pp. 387–391 vol.1.
[7] N. Weicker, G. Szabo, K. Weicker, and P. Widmayer, "Evolutionary multiobjective optimization for base station transmitter placement with frequency assignment," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 189 – 203, april 2003.

[8] J. Wong, A. Mason, M. Neve, and K. Sowerby, “Base station placement in indoor wireless systems using binary integer programming,” *Communications, IEE Proceedings-*, vol. 153, no. 5, pp. 771 –778, oct. 2006.

[9] M. Aldajani, “Convolution-based placement of wireless base stations in urban environment,” *Vehicular Technology, IEEE Transactions on*, vol. 57, no. 6, pp. 3843 –3848, nov. 2008.

[10] D. Yang, S. Misra, and G. Xue, “Joint base station placement and fault-tolerant routing in wireless sensor networks,” in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, nov. 2009, pp. 1 –6.

[11] M. Tang, “A reference station placement scheme in deployment of network real-time kinematic positioning systems,” in *Proc. International Global Navigation Satellite Systems Society Symposium*, Sydney, December 2007, pp. 123–132.

[12] ——, “QoS-aware reference station placement for regional network RTK,” *Journal of Software Engineering and Applications”*, vol. 2, no. 1, pp. 1–10, 2009.

[13] Q. Wang, K. Xu, G. Takahara, and H. Hassanein, “Transactions papers - device placement for heterogeneous wireless sensor networks: Minimum cost with lifetime constraints,” *Wireless Communications, IEEE Transactions on*, vol. 6, no. 7, pp. 2444 –2453, july 2007.

[14] B. Lin, P.-H. Ho, L.-L. Xie, X. Shen, and J. Tapolcai, “Optimal relay station placement in broadband wireless access networks,” *Mobile Computing, IEEE Transactions on*, vol. 9, no. 2, pp. 259 –269, feb. 2010.

[15] P. Nuggehalli, V. Srinivasan, C.-F. Chiasserini, and R. Rao, “Efficient cache placement in multi-hop wireless networks,” *Networking, IEEE/ACM Transactions on*, vol. 14, no. 5, pp. 1045 –1055, oct. 2006.

[16] X. Ling and K. L. Yeung, “Joint access point placement and channel assignment for 802.11 wireless lans,” *Wireless Communications, IEEE Transactions on*, vol. 5, no. 10, pp. 2705 –2711, oct. 2006.

[17] S. Sarkar, H.-H. Yen, S. Dixit, and B. Mukherjee, “Hybrid wireless-optical broadband access network (woban): network planning using lagrangean relaxation,” *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1094–1105, 2009.

[18] B. Aoun, R. Boutaba, Y. Iraqi, and G. Kenward, “Gateway placement optimization in wireless mesh networks with qos constraints,” *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 11, pp. 2127 –2136, nov. 2006.

[19] P. Li, X. Huang, Y. Fang, and P. Lin, “Optimal placement of gateways in vehicular networks,” *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 6, pp. 3421 –3430, nov. 2007.

[20] M. Tang, “Gateways placement in backbone wireless mesh networks,” *International Journal of Communications, Network and System Sciences*, vol. 2, no. 1, pp. 44–50, February 2009.