

# A Descend-Based Evolutionary Approach to Enhance Position Estimation in Wireless Sensor Networks

Vincent Tam, King-Yip Cheng and King-Shan Lui

Department of Electrical and Electronic Engineering  
The University of Hong Kong, Pokfulam, Hong Kong.  
vtam@eee.hku.hk

## Abstract

*Wireless sensor networks have wide applicability to many important applications including environmental monitoring and military applications. Typically with the absolute positions of only a small portion of sensors pre-determined, localization works for the precise estimation of the remaining sensor positions on which most location-sensitive applications rely. Intrinsically, localization can be formulated as an unconstrained optimization problem based on various distance/path measures, for which most of the existing work focus on increasing its precision through different heuristic or mathematical techniques. In this paper, we propose to adapt an evolutionary approach, namely a micro-genetic algorithm (MGA), and its variant as post-optimizers to enhance the precision of existing localization methods including the Ad-hoc Positioning System. Our adapted MGA and its variants can easily be integrated into different localization methods. Besides, the prototypes of our evolutionary approach gained remarkable results on both uniform and anisotropic topologies of the simulation tests, thus prompting for many interesting directions for future investigation.*

## 1. Our Adapted Micro-Genetic Algorithm

Evolutionary algorithms (EA) [4, 14] are local search methods capable of efficiently solving complex constrained or unconstrained optimization problems (OPT) [1]. Localization [3, 5, 6, 7, 8, 9, 10, 11, 12, 13] in wireless sensor networks is intrinsically an unconstrained OPT. However, in handling relatively sizable (with  $\geq 500$  variables) or complicated OPTs, the total computational costs can be drastically reduced by focusing the search only on a reasonably small population of chromosomes without much impact on the search efficiency. In fact, it has been reported

that MGAs can find solutions with fewer iterations than evolutionary algorithms (EAs) with larger population sizes for some problems [4]. Therefore, in this paper, we consider a micro-genetic algorithm (MGA) based on a small population size (usually  $< 30$ ) to tackle the challenging localization problems in wireless sensor networks. Basically, a variable in an OPT is usually represented by a gene in a MGA. A chromosome, consisting of all the genes, is used to denote a valuation for all the variables. As inspired by the nature, an EA or MGA basically performs a parallel local search in different parts of the search space through maintaining a population of chromosomes for iterative improvements over the successive generations. Figure 1 shows the pseudo-code

---

```
MGA( $PZ, MZ, fitness()$ )
  initialize a small Population of  $PZ$  chromosomes
  repeat
    select the best  $MZ$  chromosomes  $\in$  Population
     $Population := \emptyset$ 
    repeat
      produce {offspring} by descend-based genetic opr.
       $Population := Population \cup \{offspring\}$ 
    until ( $sizeof(Population) = PZ$ )
  until (Population is converged or resource limit is exceeded)
```

---

**Figure 1. The Convergence Procedure of our Adapted MGA**

---

of our adapted MGA as a post-optimizer for further position refinement in any localization method. Given the population size  $PZ$ , the size  $MZ$  of the mating pool and the evaluation function  $fitness()$ , our adapted MGA initially sets up a small population. In each generation, our adapted MGA selects the best  $MZ$  chromosomes according to  $fitness()$  from the current *Population* to construct the mating pool in which some genetic operators such as mutation or crossover

are applied to produce offspring to form the next generation. This “selection-and-reproduction” process is repeated until all the chromosomes have converged to the same local minima, or some predetermined resource limit is exceeded. A resource limit is usually defined in terms of CPU time or the maximum number of generations allowed. After all, our adapted MGA as a post-optimizer for any localization method aims at minimizing the best position estimates returned by the concerned localization method. Clearly, the objective function ( $fitness()$ ) for each node  $i$  in our adapted MGA for the improved APS method can be formally defined as follows.

$$fitness(x'_i, y'_i) = \sum_{j=1}^3 (\sqrt{(x'_i - x_j)^2 + (y'_i - y_j)^2} - Pdist_j)^2$$

where  $(x'_i, y'_i)$  consistently denotes the current position estimate,  $(x_j, y_j)$  is the true position of the anchor  $i$  with  $i = 1, 2, 3$  for the 3 nearest anchors as used in the improved APS method, and  $Pdist_i$  is the relative path distance measure of anchor node  $i$  with respect to the current position estimate. Clearly, the  $fitness()$  function is in fact an adapted version of the  $rel\_err()$  function described in the previous subsection with  $k$  set to 3. Thus, the function gives the summed square error of the current position estimate  $(x'_i, y'_i)$  with respect to the positions of the 3 nearest anchors and their relative path distance measures, i.e. the same as the heuristic error evaluation mechanism used in the improved APS method.

The two descend-based genetic operators for our adapted MGA are described as below.

- descend-based mutation operator: for each chromosome denoting the position estimate of node  $i$  in the current population, the operator will invoke the  $rand()$  function in C returning a probability  $p$  ranging from  $0 \dots 1$ . If  $(p > 0.5)$ , the mutation operator will generate a random point  $(x''_i, y''_i)$  in a straight line formed by the current position estimate  $(x'_i, y'_i)$  and the previous position estimate  $(x'_i, y'_i)^{old}$ . In case the newly generated mutate point  $(x''_i, y''_i)$  produces a descend in the objective value returned by  $fitness()$  when compared to that of the current estimate  $(x'_i, y'_i)$ ,  $(x''_i, y''_i)$  will replace the current position estimate  $(x'_i, y'_i)$  as the new chromosome; otherwise, the current chromosome will remain unchanged. Figure 2(a) clearly shows the working mechanism of the descend-based mutation operator used in our adapted MGA for localization. Essentially, the descend-based mutation operator will look for opportunistic improvement in each position estimate as denoted by each chromosome in the small population of our adapted MGA over successive generations.

- descend-based crossover operator: basically, the whole population of chromosomes are readily sorted in descending order according to their objective values. The crossover operator looks for any opportunistic improvement between any pair of chromosomes sequentially extracted from the sorted list of chromosomes. In this regard, we usually set the  $PZ$  to be even. In case not, the last pair can simply be formed from the last and first chromosomes of the sorted list. For each pair  $(i, i + 1)$  of chromosomes, when the probability value  $p$  as returned by  $rand()$  is  $> 0.5$ , the crossover operator will proceed to consider the mid-point as the crossover point of their position estimates. Similar to the mutation operator, in case there is any decrease in the objective value of this crossover point when compared to that of the worse chromosome  $(i + 1)$  in the pair, the chromosome  $(i + 1)$  will be replaced by the new crossover point; otherwise, nothing occurs. Figure 2(b) illustrates how the descend-based crossover operator works to improve the worse chromosome out of each pair  $(i, i + 1)$  of chromosomes through their mid-point generated. Similar to the descend-based mutation operator, this crossover operator will also be applied to the whole population over successive generations of our adapted MGA.

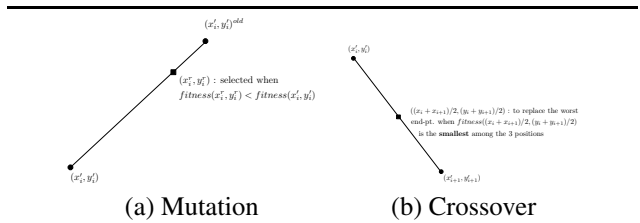


Figure 2. The Descend-Based GA Operators

It is worth noting that unlike the conventional EA, our adapted MGA does not employ any selection criterion like the Roulette Wheel mechanism [4] to remove those less favorable chromosomes from the current population. For better efficiency, we simply assume all the chromosomes will remain and be improved whenever possible by our two descend-based genetic operators over successive generations.

## 2. Experimental Results

In this section, we give an empirical evaluation of our adapted MGA and its variant using a different probability distribution to encourage more new changes in its descend-based mutation operator, namely the  $MGA^0$  and  $MGA^I$ , as *post-optimizers* to further enhance the performance of the

|            | Results for (100 nodes, 10% anchors; unit length = 1.2r) |                      |                       |                      |                      |
|------------|--|----------------------|-----------------------|----------------------|----------------------|
| Optimizers | Case 1   | Case 2               | Case 3                | Case 4               | Case 5               |
| Impr. APS  | 0.2074   | 0.6554               | 0.1879                | 0.2560               | 0.3970               |
| $MGA^0$    | <b>0.1868 (9.9%)</b>                                     | 0.6082 (7.2%)        | 0.1736 (7.6%)         | 0.2346 (8.3%)        | <b>0.3689 (7.1%)</b> |
| $MGA^I$    | <b>0.1864 (10.1%)</b>                                    | 0.6197 (5.5%)        | 0.1701 (9.5%)         | <b>0.2490 (2.7%)</b> | 0.3854 (2.9%)        |
|            | Results for (100 nodes, 10% anchors; unit length = 1.6r) |                      |                       |                      |                      |
| Impr. APS  | 0.0813   | 0.3697               | 0.0453                | 0.1598               | 0.2702               |
| $MGA^0$    | <b>0.0786 (3.3%)</b>                                     | 0.3342 (9.6%)        | <b>0.0379 (16.4%)</b> | 0.1521 (4.8%)        | 0.2529 (6.4%)        |
| $MGA^I$    | <b>0.0798 (1.8%)</b>                                     | 0.3355 (9.3%)        | <b>0.0378 (16.7%)</b> | 0.1530 (4.3%)        | 0.2635 (2.5%)        |
|            | Results for (100 nodes, 10% anchors; unit length = 2.0r) |                      |                       |                      |                      |
| Impr. APS  | 0.0534   | 0.2814               | 0.0163                | 0.0811               | 0.1742               |
| $MGA^0$    | 0.0368 (31.2%)   | 0.2631 (6.5%)        | <b>0.0110 (32.9%)</b> | <b>0.0797 (1.7%)</b> | 0.1656 (5.0%)        |
| $MGA^I$    | 0.0507 (5.1%)  | 0.2634 (6.4%)        | <b>0.0113 (31.0%)</b> | 0.0801 (1.3%)        | <b>0.1738 (0.2%)</b> |
|            | Results for (150 nodes, 10% anchors; unit length = 1.2r) |                      |                       |                      |                      |
| Impr. APS  | 0.6421   | 0.4682               | 0.3715                | 0.2604               | 0.4964               |
| $MGA^0$    | <b>0.3854 (40.0%)</b>                                    | <b>0.4619 (1.3%)</b> | 0.3133 (15.7%)        | 0.2001 (23.2%)       | 0.4524 (8.9%)        |
| $MGA^I$    | <b>0.3875 (39.7%)</b>                                    | <b>0.4625 (1.2%)</b> | 0.3187 (14.2%)        | 0.1990 (23.6%)       | 0.4692 (5.5%)        |
|            | Results for (150 nodes, 10% anchors; unit length = 1.6r) |                      |                       |                      |                      |
| Impr. APS  | 0.3817   | 0.3279               | 0.2110                | 0.1047               | 0.3281               |
| $MGA^0$    | <b>0.2011 (47.3%)</b>                                    | <b>0.3236 (1.3%)</b> | 0.1791 (15.1%)        | 0.0925 (11.6%)       | 0.2924 (10.9%)       |
| $MGA^I$    | <b>0.1998 (47.7%)</b>                                    | <b>0.3245 (1.0%)</b> | 0.1832 (13.2%)        | 0.0909 (13.2%)       | 0.2976 (9.3%)        |
|            | Results for (150 nodes, 10% anchors; unit length = 2.0r) |                      |                       |                      |                      |
| Impr. APS  | 0.2445   | 0.2378               | 0.1294                | 0.0709               | 0.2398               |
| $MGA^0$    | <b>0.1413 (42.2%)</b>                                    | <b>0.2368 (0.4%)</b> | 0.1183 (8.6%)         | 0.0596 (15.9%)       | 0.2254 (6.0%)        |
| $MGA^I$    | <b>0.1418 (42.0%)</b>                                    | <b>0.2372 (0.2%)</b> | 0.1216 (6.0%)         | 0.0593 (16.4%)       | 0.2300 (4.1%)        |

**Table 1. Performance of the Improved APS Against Those Integrated with Our Adapted MGAs on Uniform Sensor Networks**

improved APS algorithm [2] using the nearest 3 anchors on both uniform and anisotropic sensor networks. Basically, both the  $MGA^0$  and  $MGA^I$  optimizers have a fair chance of 50% to execute the descend-based mutation operator or not in each iteration. When the descend-based mutation operator is actually executed, the  $MGA^0$  optimizer makes use of a probability distribution as (0.5, 0.5) to generate a mutate with either the  $x$  or  $y$  coordinate changed for consideration whereas the  $MGA^I$  optimizer uses a new probability distribution as (0.4, 0.3, 0.3) to generate a mutate with both  $x$  and  $y$  changed,  $x$  changed or  $y$  changed respectively. Clearly, the new  $MGA^I$  optimizer is more aggressive and thus tries to encourage more radical changes, i.e. with both  $x$  and  $y$  changed, for larger improvements through the descend-based mutation operator whenever possible. To demonstrate the effectiveness of our adapted MGAs, we generated 5 random instances of uniform or C-shape topologies for wireless sensor networks to evaluate the performance of both the improved APS and the improved APS integrated with our adapted  $MGA^0$  or  $MGA^I$  as the post-optimizer for position refinement. For all the test cases, the number of anchors is always set as 10% of the total number  $N$  of sensors, where  $N = 100$  or 150 in our simulation tests.

Table 1 compares the performance of our originally improved APS and the improved APS integrated with our adapted MGAs in terms of the mean position error on different combinations of the total number  $N$  of sensor nodes as 100 or 150, with the physical dimension of the unit square for spreading the sensors as 1.2r, 1.6r, 2.0r respectively on uniform networks. Among the highlighted improvements shown in Table 1, the largest improvement achieved by

$MGA^0$  is 47.3% whereas the smallest is 0.4%, giving an overall average of 13.5% for all the test cases on uniform sensor networks. On the other hand, for the  $MGA^I$  optimizer, the largest improvement is 47.7% with its lowest at 0.2%, producing a relatively lower overall average of 11.5% for all the uniform test cases. This clearly demonstrates the effectiveness of our adapted MGAs as post-optimizers to further enhance the position estimation of the improved APS and possibly other localization methods. In general, the  $MGA^0$  optimizer tends to be more stable in performance when compared to that of the  $MGA^I$  optimizer on these uniform test cases. This trend is more obvious in the test cases with 100 nodes. For instance, with (100 nodes, 10% anchors; unit length = 1.2r), the variation in % of improvements for the  $MGA^0$  optimizer ranges from 7.1% to 9.9% whereas that for the the  $MGA^I$  optimizer varies from 2.7% to 10.1%. Similarly, for the next parameter setting with (100 nodes, 10% anchors; unit length = 1.6r), the  $MGA^0$  optimizer gives its % of improvements ranging from 3.3% to 16.4% whereas the the  $MGA^I$  optimizer gives a wider range of % of improvements from 1.8% to 16.7%. As explained previously, this is due to the different probability distribution used in the mutation operator of the  $MGA^I$  optimizer that always tries to encourage more drastic changes for opportunistic improvements during the search, thus giving a relatively wider range of overall performance. However, the variation in % of improvements between the two adapted MGAs tend to be less obvious in the test cases with 150 nodes. For example, the range of % of improvements for  $MGA^0$  with (150 nodes, 10% anchors; unit length = 1.2r) is 1.3% — 40.0% which is fairly close to the range of 1.2% — 39.7% as obtained by the  $MGA^I$  optimizer. The specific reason(s) for this relatively stable improvements as attained by both of our adapted MGAs when the number of nodes increase prompts for further investigation. A possible explanation may be attributed to a wider choice of anchors available with a larger number (15 versus 10) of anchor, thus giving rise to estimated distance measures of higher precision that incur relatively less estimation errors to the overall computation. These resulting estimated distance measures of higher accuracy may help to increase the performance of both adapted MGAs to a certain extent that provides no extra benefit to our more aggressive mutation operator used in the  $MGA^I$  optimizer. This is obvious from the averaged improvements attained by both adapted MGAs as consistently over 13% with their best improvements at around 40% or more for every parameter setting with  $N = 150$  whereas the averaged improvements for the test cases with  $N = 100$  are consistently below 9% except for one specific result obtained by the  $MGA^0$  optimizer with (100 nodes, 10% anchors; unit length = 2.0r). Such specific result may be due to the probabilistic nature of our stochastic  $MGA^0$  optimizer. We would conduct more

experiments with a careful analysis to verify about this specific result in the future study.

Both of our adapted MGAs are efficient to run as post-optimizers to enhance the position estimates for uniform sensor networks. The non-linear optimizer available in the Matlab system requires minutes or even hours in CPU time to produce useful results for the improved APS method whereas our adapted MGAs, implemented in C, requires only a few to several tens of CPU seconds on the average. Precisely, the averaged CPU times as measured by the relevant timing function in the Matlab Version 7.2 system for executing our adapted  $MGA^0$  optimizer are 9.69 seconds and 21.06 seconds respectively for  $N = 100$  and 150 on uniform sensor networks. For the  $MGA^I$  optimizer, the averaged CPU times are 9.38 seconds and 19.75 seconds respectively when  $N = 100$  and 150 on uniform networks. This clearly demonstrates the efficiency of our adapted MGA proposals on uniform sensor networks.

Lastly, on anisotropic sensor networks, the largest improvement achieved by  $MGA^0$  is 66.3% whereas the smallest is 0.5%, giving an overall average of 18.6%. For the  $MGA^I$  optimizer, the best improvement is 33.6% whereas the worst is 0.2%, producing an overall average of 11.0%. Clearly, the  $MGA^0$  optimizer excels the  $MGA^I$  optimizer in both the best and overall average of improvements, thus demonstrating the effectiveness of our relatively stable  $MGA^0$  optimizer for the anisotropic networks.

### 3. Conclusion

Wireless sensor networks are widely applicable to many practical applications including environmental monitoring, military applications, disaster management, etc. in which sensors may need to know their geographical locations. Extending from our previous work [2], we proposed in this paper a generic micro-genetic algorithm (MGA) that make use of two key genetic operators, namely the descend-based mutation and crossover operators, aiming at opportunistic decreases in the objective values obtained for the new mutate out of the current position estimate, or the crossover point for any pair of existing chromosomes over successive generations for localization. Our simulation results clearly demonstrate the effectiveness of our adapted MGA proposals as efficient post-optimizers in reducing the average estimation error maximally to around half of those results attained by the originally improved APS method on both uniform and anisotropic sensor networks.

This work opens up a number of interesting directions for future exploration. First, we should conduct more experiments and careful analysis to explain in detail about the general increase in the percentages of improvements when the total number  $N$  of sensor nodes increases, and more importantly the reason(s) accountable for the specific

changes across the different values of  $R$  at the same  $N$  for both adapted MGA proposals on the uniform or anisotropic sensor networks. Besides, it should be interesting to study about the integration of our generic MGA proposal into other localization algorithms. Lastly, we may consider to employ more sophisticated candidate selection criteria so as to further increase the overall accuracy of the two descend-based genetic operators for position estimation.

### References

- [1] D. P. Bertsekas. *Nonlinear Programming (2nd Edition)*. Athena Scientific, 1999.
- [2] K. Cheng, V. Tam, and K. Lui. Improving aps with anchor selection in anisotropic networks. In *Proceedings of the International Conference on Networking and Services (ICNS'05)*, October 2005.
- [3] L. Doherty, K. Pister, and L. Ghaoui. Convex position estimation in wireless sensor networks. In *IEEE Infocom*, 2001.
- [4] G. Dozier, J. Bowen, and D. Bahler. Solving small and large scale constraint satisfaction problems using a heuristic-based microgenetic algorithm. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1994.
- [5] J. Hightower and G. Boriello. Location systems for ubiquitous computing. In *IEEE Computer*, volume 34, pages 57 – 66, August 2001.
- [6] X. Ji and H. Zha. Sensor positioning in wireless sensor networks using multidimensional scaling. In *IEEE Infocom*, 2004.
- [7] H. Lim and J. C. Hou. Localization for anisotropic sensor networks. In *IEEE Infocom*, 2005.
- [8] D. Niculescu and B. Nath. Ad hoc positioning system (aps). In *IEEE Globecom*, pages 2926 – 2931, 2001.
- [9] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. In *Communications of the ACM*, volume 43, pages 51 – 58, 2000.
- [10] C. Savarese, J. Rabaey, and K. Langendoen. Robust positioning algorithm for distributed ad-hoc wireless sensor networks. In *USENIX Technical Annual Conference*, 2002.
- [11] A. Savvides, H. Park, and M. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.
- [12] Y. Shang and W. Ruml. Improved MDS-based localization. In *IEEE Infocom*, 2004.
- [13] Y. Shang, W. Ruml, and Y. Zhang. Localization from mere connectivity. In *ACM MobiHoc*, 2003.
- [14] V. Tam and P. Stuckey. Improving evolutionary algorithms for efficient constraint satisfaction. In *The International Journal on Artificial Intelligence Tools*, volume 28, pages 363 – 383, 1999.