4th International Conference on Electrical and Computer Engineering
ICECE 2006, 19-21 December 2006, Dhaka, Bangladesh

GRADING A MICROCONTROLLER: A TRAVELLING SALESMAN PROBLEM APPLICATION

Nitesh Goyal
Tata Consultancy Services Ltd., India
5B-80, TCS, Yantra Park, Thane, Maharashtra, India
Nitesh.Goyal.84@gmail.com

ABSTRACT

One of the most financial and temporal investments in embedded systems business is the decision involved in selecting the right components, in particular, the microcontroller. Each microcontroller is made for a particular application. Since, with time, the applications get more complex, the lifespan of existing microcontrollers gets reduced. Currently industry chooses microcontrollers in an unsystematic manner, thus reducing the possibilities of employing a perfectly suited microcontroller, albeit originally meant for a different application. This paper solves this dilemma by using a modified version of Travelling Salesman Problem to grade microcontroller and find out to what extents it would be successful when used in other applications(except for the one it was designed for). This paper starts by mentioning a few common applications of embedded systems and what differentiates them. Then the proposed grading process is described and correlated with TSP, solvable by Genetic Algorithms.

1. INTRODUCTION

1.1. Application Centric Demands

Embedded systems in various applications are routinely expected to maintain 100% reliability while running continuously for long periods of time, sometimes measured in years [1]. Different embedded system applications have different demands from components of the devised embedded systems. For example, the following applications need to specifically emphasize more on particular requirements as listed below:

- ATM architectures must be Reliable 24 x 7 but do not have much interfacing requirements.
- Cellular Telephone architectures have to be Power saving, Cost effective but do not need much Legacy Support
- Engine Controllers have to be Performance intensive and should withstand varied working temperature conditions.
- Toys should be cheap and relatively have less performance and reliability requirements.

1.2. Travelling Salesman Problem

TSP is a NP-hard combinatorial problem, easier to state than solve. Given a collection of cities and the cost of travel between each pair of them, TSP, is to find the cheapest way of visiting all the cities and returning to the starting point. Mathematically, In TSP we are given n points (or "cities") c(1). . . c(n) and a positive distance d[c(i), c(j)] for each distinct pair of cities. Our goal is to find an ordering Π, or tour, of the cities that minimizes the length of the tour, l given by the following equation:

$$l = d[c_{\Pi}(n), c_{\Pi}(1)] + \sum (i=1 \text{ to } n-1)\ d(c_{\Pi}(i), c_{\Pi}(i+1))$$

1.3. Problem Definition

Thus, problem at hand can be defined as: **To generate a Microcontroller-Application Mapset with graded microcontroller suitability**. To do the same we need to consider the microcontroller as a salesman which "goes to each application" and finds out the lowest possible distance (described in section 2). In case multiple cities are found, these are compared amongst each other to find out a unique answer (described in section 3). So, to convert the problem to TSP, we have to do the following:

1. Reduce Applications to "Cities" (section 2)
2. Find the Distance Function (section 3)
3. Find the Fitness Function (section 4)
4. Implement GA to solve TSP problem (section 5)

2. MEASURING APPLICATIONS

2.1. Applications and Parameters

Owing to their unique requirements, the applications have been categorized into 12 major sets called APPLICATIONS. They are:

List1:

Defence, Space, Toys, Automobiles, Consumer Electronics, Wireless Devices, Medical Imaging, Entertainment Equipment, Control Devices/Battery, Manufacturing/Robotics, N/W Equipment, Others.

"Others" is a soft- application which means that this can be customized for much more specific results. Being a hardware device, it is relatively easier to judge the performance of the chip. Various parameters that can describe the microcontroller can be compared amongst each other to reach to the conclusion. For the microcontroller differentiation and to maintain a high level of uniqueness and accuracy, 30 parameters have been considered. These 30 parameters have also been categorized into 9 sets. Each set represents closely resembling parameters which reflect similar capabilities. The CATEGORIES are:

List2:

1. CPU PERFORMANCE {Max Xtal Speed, MIPS}
2. MEMORY PERFORMANCE
{On Chip RAM, On Chip XRAM, On Chip FLASH, On Chip ROM, On Chip EPROM, ROM Lock}
3. POWER
{Power supply, Pulse down mode, Idle mode}
4. LEGACY {Packages available}
5. DSP {A/D Channels, A/D Bits, D/A Channels, D/A Bits, PCA, PWM}
6. NETWORKING {UART, USART, SPI, ICRT}
7. RTOS
{Timer/ Counters, Real Time Clock, Watchdog Timer}
8. I/O {I/O Ports, Interrupt sources, Interrupt bits}
9. RELIABILITY {On Chip Debug, Brown out detect}

2.2. Application Specifications

Each APPLICATION has been given 4 CATEGORIES in decreasing priority according to their nature and requirements. They are enlisted as below:

1. DEFENCE: Reliability, CPU Performance, Memory Performance, Power
2. SPACE: Reliability, Power, CPU Performance, Memory Performance
3. TOYS: Legacy, Power, Reliability, RTOS
4. AUTOMOBILES: Reliability, I/O, RTOS, Power
5. CONSUMER ELECTRONICS: Power, Reliability, Legacy, RTOS
6. WIRELESS DEVICES: Input/Output ports, N/W, Reliability, Memory Performance
7. MEDICAL IMAGING: DSP, RTOS, CPU Performance, Memory Performance
8. ENTERTAINMENT EQUIPMENT: DSP, I/O, Power, Legacy
9. CONTROL DEVICES/BATTERY: RTOS, DSP, Reliability, CPU Performance
10. MANUFACTURING/ROBOTICS: RTOS, I/O, Legacy, Reliability
11. NETWORKING DEVICE: N/W, Reliability, I/O, Power
12. OTHER: CPU Performance, Memory Performance, DSP, RTOS

In section 2.1, "by customizing", we meant that the priorities can be reassigned to various categories.

Table1. Weights awarded to categories for each application

cp	mp	power	reliability	legacy	io	dsp	rtos	nw	application
3	2	1	4	0	0	0	0	0	DEFENCE
3	1	2	4	0	0	0	0	0	SPACE
0	0	3	2	4	0	0	1	0	TOYS
0	0	1	4	0	3	0	2	0	AUTOMOBILES
0	0	4	3	2	0	0	1	0	CONSUMER ELECTRONICS
0	1	0	2	0	4	0	0	3	WIRELESS DEVICES
2	1	0	0	0	0	4	3	0	MEDICAL IMAGING
0	0	2	0	1	3	4	0	0	ENTERTAINMENT EQUIPMENT
1	0	0	2	0	0	3	4	0	CONTROL DEVICES/ BATTERY
0	0	0	4	3	2	0	1	0	MANUFACTURING/ ROBOTICS
0	0	1	3	0	2	0	0	4	NETWORKING EQUIPMENT
4	3	0	0	0	0	2	1	0	OTHERS

WEIGHTS: weights from 0 to 4 are assigned to 4 CATEGORIES in deceasing order to signify priority to each APPLICATION. A representation of weights distributed for different categories can be seen above in Table1.

3. ADAPTING TSP TO MICRO CONTROLLER GRADING PROBLEM

In our problem, since the distance of one application from another is equal in both the directions, thus,

$$d[c_{\Pi}(i), c_{\Pi}(i+1)] = d[c_{\Pi}(i+1), c_{\Pi}(i)] \quad (1)$$

So, the number of paths required will be n! / 2.

The application value is the weighted sum of category values. And category values are the sum of percentile of parameter values. We start by finding the relative suitability of a particular application to the microcontroller by finding out the application value for each application..

Algorithm1: Calculation of cat_val of each category

This algorithm calculates the value of each category for each application:

1. Start
2. //Calculate the percentile value of each parameter by dividing with maximum parameter value available.

For (i=0 to 29)

percentile_par_val [i] = par_val [i]/ max_par_val [i]

3. //Calculate the value of each specification

spec_val[j] = $\sum$ (percentile_par_val [k])

Where

j = 0 to 8, and

k= {(1.1, 1.2), (2.1, 2.2…2.6)... (9.1, 9.2)}

4. //Calculate the application value

app_val = $\sum$ (spec_val [j] * w (j)) (2)

Where j = 0 to 8

5. End

Since, we have got all the application values (app_val) for each APPLICATION for this microcontroller, one may argue that it is enough and

now the maximum value can be easily selected and the corresponding APPLICATION can be declared to be the best fit. This has many limitations, most important amongst them maybe:

1. It is possible to have equal app_val for different APPLICATION. Hence, the question of "Best Fit" still remains unanswered.

2. This does not take into account, how really each comparing application is different from the other application.

Thus, a grading mechanism which "fits" the microcontroller for each application and also grades applications according to their suitability is needed. It can be solved by modifying the TSP in such a way that the route does not end at the starting city (application) and each city (application) is traversed once.

Moreover, to handle limitation 1, the first application is chosen such that it is the most ill fitting application. So from equation (1), the number of possible routes would be (n-1)! /2. Considering, in our case, n (applications) = 12.

Thus, total number of paths = 11!/2=19958400. Since, travelling salesman searches for a Hamilton outline of minimum length; we should leverage the progress we have made in solving TSP using heuristic algorithms. And, Genetic Algorithm has proven to give robust solution to TSP. [2]

4. ADAPTING GA TO MICRO CONTROLLER GRADING PROBLEM

Each chromosome is of length equal to number of applications and is created by a random sequence generator. Another random number generator generates the Mutation pivot about which the mutation occurs. Then it goes through each chromosome trying to find the least distance amongst the applications.

Algorithm 2: Determining Fitness Function for Grading for application

This algorithm bubbles the best application to the end of a gene sequence after multiple mutations and applications get arranged in increasing order of suitability through the chromosome. So, the following changes need to be made in finding out the Fitness Function.:

1. We have to take into consideration the heuristic of first gene chosen in the chromosome by comparing with the worst gene possible in the gene set. So, we calculate the penalty p for first application chosen in this gene from equation 2 in Algorithm 1:

a = the app_val for this microcontroller for first application

a_{max} = the maximum app_val for this microcontroller

$$p = a/a_{max} \quad (3)$$

2. Calculate distance b/w each pair of successive genes of this chromosome,

So for chromosome ci and gene g(j) and g(j+1)

We will get the distance by:

2.1. Summing the category values of uncommon categories, k_uc, of consecutive genes j and j+1

$$d1 = \sum(cat\_valg(j,k\_uc) + cat\_valg(j+1,k\_uc))$$
$$\forall (j, k\_uc), cat\_valg(j,k\_uc) \neq cat\_valg(j+1,k\_uc)$$

2.2. Summing difference of category values of common categories, k_c of consecutive genes j & j +1.

$$d2 = \sum(cat\_valg(j,k\_c) - cat\_valg(j+1,k\_c))$$
$$\forall (j, k\_c), cat\_valg(j,k\_c) = cat\_valg(j+1,k\_c)$$

Now,

Total no. of categories = 9

Total no. of prioritized categories = 4

So, the total distance, $d = d1/9 + d2/4$ (4)

3. Since, Fitness function, $f \sim 1/d$, and $f \sim 1/p$. So, from equation (3) and (4), we get:

$$f = k * 1/d * 1/p$$

For, sake of computational ease, putting k = 1, we get

$$f = 1/d * 1/p = 1/dp \quad (5)$$

5. IMPLEMENTATION

The datasheets for microcontrollers were obtained from Keil database. [3] The algorithm was implemented using GAlib library. [4] In GAlib, the sample data structure is called a GAGenome. The data structure that has been used is GAArrayGenome which is derived from the base GAGenome class and a data structure class called GAArray class. The Steady state version of GA that uses overlapping populations and is similar to algorithms described by DeJong [5] has been used to implement the solution to this problem. The amount of overlap between generations by specifying the Replacement parameter (set to 30%) to allow for best solutions of the previous generation to pass onto next generation. The chromosomes are implemented as GAArrayGenomes, 1D string arrays, of length equal to the total number of applications which is 12. And the fitness function used for the chromosome was from equation (5).

Another array is defined for each genome at run time which keeps a track of the flag if a city has been visited more than once in a tour or not.

6. RESULTS

6.1 Result Set 1

The numbers in Generation 10 listed below represent APPLICATION in the order they have been mentioned in List 1 and following is a brief gist of the microcontroller sample of Atmel AT89C51ID2:

The Atmel AT89C51ID2 is an 80C52-compatible High-Speed Microcontroller with 48 I/O Lines, WDT, 3 Timer/Counters, 16-bit PCA, PWM, Dual DPTR, SPI, UART, Integrated Power Monitor, 10 Interrupts/4 Priority Levels, 64K Bytes on-chip Flash ROM, ISP (In-System Programming), 256 Bytes on-chip RAM, 1792 Bytes XRAM.

Following is the tenth generation stabilized population of 10 chromosomes for this microcontroller.

GENERATION 10
5 4 3 6 8 10 11 7 1 2 12 9 -->1.247186
5 4 3 6 8 10 11 7 1 2 12 9 -->1.247186
5 4 3 6 8 10 11 7 1 2 12 9 -->1.247186
5 4 3 6 9 10 11 7 1 2 12 8 -->**1.201908**
8 4 3 6 9 11 10 5 1 2 12 7 -->1.249686
8 4 3 6 9 11 10 5 1 2 12 7 -->1.249686
5 4 3 6 8 10 11 7 1 2 12 9 -->1.247186
5 4 3 6 8 10 11 7 1 2 12 9 -->1.247186
5 4 3 6 8 10 11 7 1 2 12 9 -->1.247186
8 10 4 3 7 6 2 9 12 11 1 5 -->1.229130

As can be seen here, in this generation, the fourth chromosome has the smallest relative distance between the various applications. This reflects how closely various applications are related to each other for this particular microcontroller specifications.

Most of them will have the same starting application because of the PENALTY, p involved in starting with the farthest application. This happens because as we go down the generations only the least PENALTY levied chromosomes will be chosen.

Thus, this particular microcontroller is best suited for 8 i.e. Entertainment-Equipment.

6.2 Result Set 2

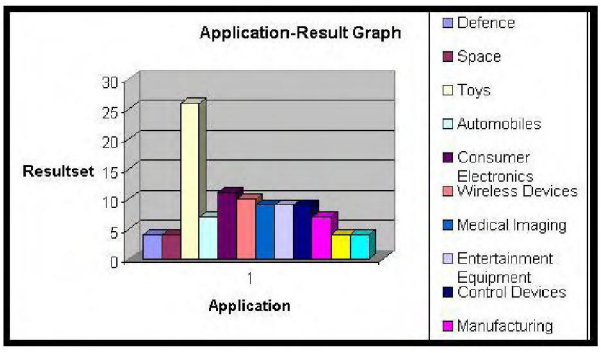

Fig. 1 APPLICATION RESULT SET GRAPH: This shows the percentage of the number of microcontrollers that were mapped to each application. "Toys" being the application to which maximum microcontrollers can be mapped.

Fig.1 is a sampled representation of 300 microcontrollers considered in the experiment. The graph indirectly signifies that the suggested solution works because as depicted, most of the microcontrollers would be suitable to be used as Toys because they have all the performance capabilities required.

Since, "Toys" is a very generic application where almost all the microcontrollers can be used, so the number of microcontrollers graded maximum (around 200) for Toys. On the other hand, for applications like defence and space, very few microcontrollers would prove to be suitable owing to relatively enormous requirements demanded by these applications. Thus, not more than 40 microcontrollers were found suitable for them.

As shown, the Result 1 shows an effective way of grading and finding suitable application while Result 2 confirms the ability to segregate the input sample data provided vis-à-vis the application set.

7. FUTURE IMPROVEMENTS

As evident in the results, quite a large no. of the duplicate chromosomes is being created in a single generation which decreases tool's efficiency. TSP was optimized first by Whitley [6] using modified edge recombination. Complexity and population size of this problem is similar to a280 of TSPLIB [7] (Ludwig's drilling problem of 280), optimal gene order technique can be implemented [8], with no. of best individuals to be marked = 20 and no. of best individuals from which overlap vectors are abstracted = 60. Due to insufficient time this could not be implemented.

8. REFERENCES

[1] Philip J Koopman, Jr.: Proceedings of the International Conference on Computer Design (ICCD 96).

[2] Novikov F. A., Discrete mathematics for programmers, St. Piter.: Piter, 2000.-304 p.:pic.

[3] Datasheets of Microcontrollers by major manufacturers: Keil Database, www.keil.com and manufacturer's websites

[4] GaLib, Matthew Wall, MIT (http://lancet.mit.edu/ga/).

[5] K. de Jong. Genetic algorithms: A 10 year perspective. In Proceedings of the first International Conference on Genetic Algorithms and their Applications, pages 169 -- 177, 1985.

[6] D.Whitley, T.Starkweather, and D.Shaner: "Travelling salesman and sequence scheduling: Quality solutions using genetic edge recombination" in Handbook of GA, 1990.

[7]http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/ : TSPLIB Home page.

[8] Jun Lu, Boqin Feng, Bo Li: "Finding the Optimal Gene Order for Genetic Algorithm", Proceedings of World Congress on Intelligent control and Automation, 2004.