ECE 6740: Advanced Embedded Systems

Homework #1

Directions for this homework: Please use this Microsoft Word document and change the font color for your responses to BLUE. Make space in line with the questions to place or paste your answers/figures/MATLAB commands. Upload your completed homework file to Moodle.

Q1. Find an optimal mapping for the following 4 objects (tasks) to the 3 blocks (implementation targets) listed below with respect to two objectives: (1) minimize execution time and (2) minimize power by solving an integer linear program (ILP).

Table Q1.1 below shows the execution time measured by executing each task on an ARM core, a DSP, and using HLS to create digital circuits that implement each task to obtain estimated execution resources. Assume that there are no bus bottlenecks interconnecting any of these implementation targets.

Table Q1.1: Execution time **(ET)**

| **Execution time (normalized units)** | Task 1 | Task 2 | Task 3 | Task 4 |
| --- | --- | --- | --- | --- |
| Block 1: ARM | 5 | 10 | 15 | 5 |
| Block 2: DSP | 10 | 10 | 10 | 10 |
| Block 3: PL | 15 | 15 | 5 | 5 |

Table Q1.2 shows the power required for each mapping.

Table Q1.2: Power required **(P)**

| **Power (normalized units)** | Task 1 | Task 2 | Task 3 | Task 4 |
| --- | --- | --- | --- | --- |
| Block 1: ARM | 3 | 3 | 3 | 2 |
| Block 2: DSP | 2 | 1 | 3 | 2 |
| Block 3: PL | 1 | 2 | 2 | 3 |

Use the following overall cost function, *OCF*, for each binding of task *i* to block *k*:

= 3 \* + 2 \*

Formulating this as an ILP problem, we can indicate task *i* is bound to block *k* using a binary representation *x* below:

where 0 represents no binding and 1 represents a binding

minimize f =

subject to

PART A: Solve this problem by expanding the function and identifying, by inspection, what binding you would choose to minimize *f* and what overall cost, *OCF*, would result from your choice. Show your work.  
Applying a scalar of 3 and 2 to tables Q1.1 and Q1.2, respectively we get the following

Table Q1.1 (Scaled 3): Execution time **(ET)**

| **Execution time (normalized units)** | Task 1 | Task 2 | Task 3 | Task 4 |
| --- | --- | --- | --- | --- |
| Block 1: ARM | 15 | 30 | 45 | 15 |
| Block 2: DSP | 30 | 30 | 30 | 30 |
| Block 3: PL | 45 | 45 | 15 | 15 |

Table Q1.2 (Scaled 2): Power required **(P)**

| **Power (normalized units)** | Task 1 | Task 2 | Task 3 | Task 4 |
| --- | --- | --- | --- | --- |
| Block 1: ARM | 6 | 6 | 6 | 4 |
| Block 2: DSP | 4 | 2 | 6 | 4 |
| Block 3: PL | 2 | 4 | 4 | 6 |

**Iterate over K**

Taking the lowest execution time in each column provides the lowest OFC because the ET is much larger than P values. When a tie occurs, take the lowest P value.

Task 1: Block 1 => (15 + 6) = **21**

Task 2: Block 2 => (2 +30) = **32**

Task 3: Block 3=> (15 + 4) = **19**

Task 4: Block 1=> (15 + 4) = **19**

**Min f = 21 + 19 + 19 + 32 = 91**

PART B: Using MATLAB’s *intlinprog* function, formulate the problem above and solve it using MATLAB. Submit your MATLAB steps and a screenshot of the resulting *x* and *fval*. You can find direction on this function at:

<https://www.mathworks.com/discovery/integer-programming.html>

and a helpful example at

<https://www.mathworks.com/help/optim/ug/mixed-integer-linear-programming-basics.html>

ZIP CONTAINS SCREENSHOT

ET = [5,10,15,5, 10,10,10,10, 15,15,5,5]; % define matricies

P = [3,3,3,2, 2,1,3,2, 1,2,2,3]; % define matricies

f = 3\*ET(:)' + 2\*P(:)'; % Apply the equation into a single vector

numCol = 12; % columns in f

intcon = 1:numCol; % define values to change

lb = zeros(numCol,1)'; % set lb to 0s

ub = ones(numCol,1)'; % limit to binary

blocks = 3;

tasks = 4;

% construct the Aeq

Aeq = zeros(tasks,numCol);

for n = 1:tasks

for k = 1:blocks

scale = n+(tasks\*(k-1));

Aeq(n, scale) = 1;

end

end

beq = [1,1,1,1];

[x, fval] = intlinprog(f, intcon, [], [],Aeq, beq, lb, ub);

x\_i = find(x == 1);

display("x index selected: " + x\_i);

display("Index values: " + f(x\_i));

display("Fval: " + fval);

PART C: Add one more constraint to the ILP problem above where the total Power required must be less than or equal to 7 units as shown below. Use MATLAB to solve the ILP.

minimize f =

subject to

and

(ZIP CONTAINS PART\_C SCREENSHOT)

With the new constraint, the solution is **fval: 104** | **x = {4,5,6,11}** for the sum matrix. Where each element is a calculated OCF

***ET = [5,10,15,5, 10,10,10,10, 15,15,5,5]; % define matricies***

***P = [3,3,3,2, 2,1,3,2, 1,2,2,3]; % define matricies***

***f = 3\*ET(:)' + 2\*P(:)'; % Apply the equation into a single vector***

***numCol = 12; % columns in f***

***intcon = 1:numCol; % define values to change***

***lb = zeros(numCol,1)'; % set lb to 0s***

***ub = ones(numCol,1)'; % limit to binary***

***blocks = 3;***

***tasks = 4;***

***% construct the Aeq***

***Aeq = zeros(tasks,numCol);***

***for n = 1:tasks***

***for k = 1:blocks***

***scale = n+(tasks\*(k-1));***

***Aeq(n, scale) = 1;***

***end***

***end***

***beq = [1,1,1,1];***

***A = P(:)';***

***b = 7;***

***[x, fval] = intlinprog(f, intcon, A, b,Aeq, beq, lb, ub);***

***fprintf("x index selected: ");***

***x'***

***fprintf("p values: ");***

***(x.\*P(:))'***

***fprintf("fValues :");***

***(3.\*(x.\*ET(:)) + 2.\*(x.\*P(:)))'***

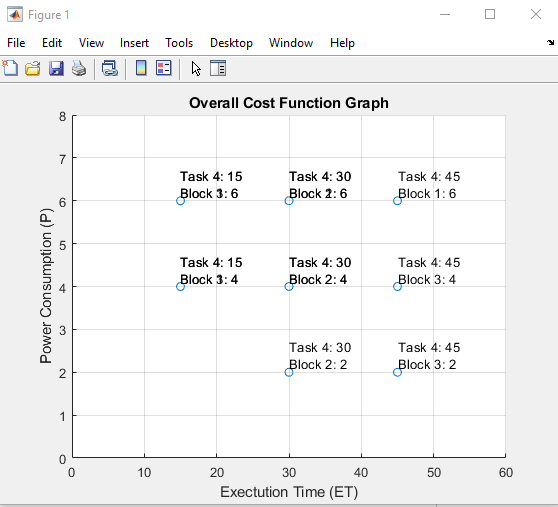
***fprintf("Fval: " + fval + '\n');***

Q2. Instead of using an apriori overall cost function to map the multi-dimensional problem of optimizing execution time and power onto one single-valued function to optimize from #Q1, you want to inspect Pareto-optimal values to see what tradeoffs between execution time and power are available to you to present to your management team at the next design meeting.

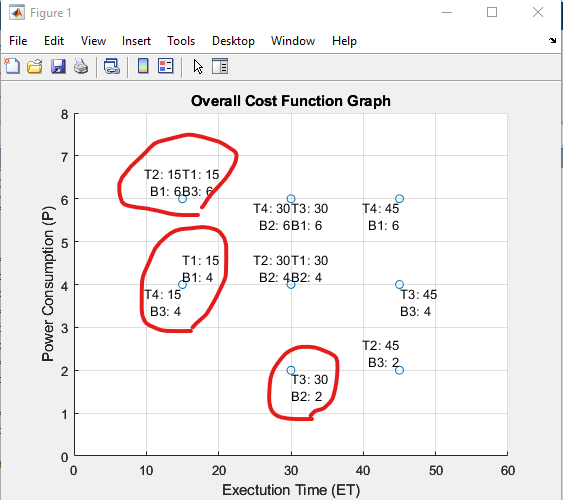
PART A: How many possible combinations of bindings can be made from #Q1?

12 possible bindings. blocks\*tasks

PART B: Graph all of the possible bindings on a 2-D graph depicting execution time and power. Use MATLAB.



PART C: Identify, by inspection, which bindings form a pareto-optimal front; either circle them on your graph or list them. How many are there?



The following 5 points form the Pareto Front: Task2/Block1; Task1/Block3; Task4/Block3; Task1/Block1; Task3/Block2

PART D: In this problem, there are a limited number of points and dimensions (only 2) from which it is feasible to identify the bindings that form a pareto-optimal front by inspection. For problems with larger dimensions or more objects or blocks, you would need to create an algorithm to determine the pareto-optimal front. Explain, in your own words, how the Hypervolume Indicator can help you to automate a process to determine a pareto-optimal front.

By using the Hypervolume Indicator, we can define our object space, area, and seek to find the combination of bindings that maximize this space. This process consists of systematically removing points from our volume and redrawing the area. We can store each area in some vector and find whichever binding removal resulted in the smallest decrease. We can then continue this process until we’ve achieved the appropriate number of bindings and a maximum objective space!

Q3. After performing Q1 and Q2 using your model in Tables Q1.1 and Q1.2 and presenting it to your management team, you made a design choice, and you and your team developed the product, and it is out on the market. Additionally, you have received a promotion to Senior Engineer because of your systematic approach to design optimization that led to a product that had more battery life and ran faster (both!) than the competition’s ad-hoc approach.

As a Senior Engineer, you know that it is important to not rest on your success because you suspect that the competition will catch up. So, to stay far ahead of them, you go back to working on your model and introduce some more sophisticated parameters to your model.

In addition to Tables Q1.1 and Q1.2, you add the following details by estimating them at functional and architectural levels:

* If Task 1 and Task 2 are not bound to the same object, there is an execution time increase of 1 unit because of shared memory between the two tasks.
* If Task 2 and Task 3 are not bound to the same object, there is an execution time increase of 2 units because of shared memory between the two tasks.
* If Task 1 is bound to the Block 2 (DSP) and Task 3 is bound to Block 3 (PL), there is an execution time increase of 2 units because of bus latency between the DSP and PL.

You also add cost to the model, which is a 3rd dimension added to your previous 2-D model of execution time and power:

* If Block 1 is allocated, it costs $10.
* If Block 2 is allocated, it costs $15.
* If Block 3 is allocated, it costs $22.

By adding this new information, it would be very difficult to formulate an ILP problem.

Part A: Discuss why it would be very difficult to formulate an ILP problem from this.

In addition to a third dimension, the new constraints make it difficult to define the intlinprog function in matlab. We run into the “difficult sudoku” problem. We could spend a lot of time brute forcing all possible options that fit our constraints, but this is terribly inefficient. With this increase in complexity, we could also experience significantly increased computation time

Instead, you are going to switch to an evolutionary algorithm where you can write a function to determine a bindings’ fitness that can (and will) take multiple steps to calculate and can consider which bindings should comprise the next generation by determining which ones increase the Hyperplane Indicators area. You are planning on using a genetic algorithm.

Part B: How will you represent a valid binding in your genetic algorithm?

Solution Design

There will be 4 genes in each chromosome, 1 for each task. The gene will be an index in the 12 element vector which contains the 12 OCF values. Assuming there is no scalar for $ cost (C), the new OCF would be => OCF = 3 \* ET + 2 \* P + C

Solution Layout

[1,5,9][2,6,10][3,7,11][4,8,12] will be the possible solutions. The fitness function will be the sum of each OCF selected with the additional constraints.(Each gene will be one of the 3 blocks)

NOTE: With the addition of the 3rd dimension (Cost ‘C’), we will need to ensure our fitness function can distinguish between the blocks. E.g genes 1,2,3,4 => Block 1; genes 5,6,7,8 => Block 2 etc.

Part C: How will you perform crossover to generate two valid children from two parent bindings?

Parent 1 and Parent 2 will each swap their block selections for genes 1|2 and 3|4 respectively. If they share the same block selections

Example Crossover:

Parent 1: [1][6][11][4]

Parent 2: [1][2][3][12]

Breeding result:

Child 1: [1][6][3][12]

For now, I’m arbitrarily deciding to takes the first 2 tasks blocks from parent 1 and the rest from parent 2.

Part D: How will you perform mutation to generate a valid mutant binding from a randomly selected binding in the population?

Since we know the possible pool of block choices for any given gene, we can simply select a random gene(s) and give it a new value form the pool. The number of genes selected could be 1-2.

Example Mutation:

Solution 1:[1][6][11][4]

Solution 2:[5][10][7][4]

Mutation Result

Mutation 1: [3][6][7][4]

Mutation 2: [5][6][11][4]

Q4. Super cool extra credit.

Do you want to be super cool? 😊

Part A: Use MATLABs *ga* function to use a genetic algorithm to find a small set of optimal solutions from Q3. List some of the optimal solutions and their fitness values. What would you select? Feel free to “estimate” (make up) metrics if needed. If you do so, indicate here what you added.

No new metrics were added. We used the new OCF function as the initial cost function before constraints were applied.  
OCF => 3\*ET + 2\* P + C

Since the total possible solutions is relatively small, I chose a population of 20.

The best solution was:  
[T1, T2, T3, T4] Idx’s [1,2,11,4] => [Block 1, Block 1, Block 3, Block 1] : Fitness = 149

Here are the all the solutions:  
**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 7 4 Score: 159**

Solution: 1 10 11 4 Score: 173

**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 11 4 Score: 149**

**Solution: 1 2 11 4 Score: 149**

Solution: 1 2 11 8 Score: 169

Solution: 5 2 7 4 Score: 178

Solution: 5 6 3 4 Score: 188

Solution: 5 2 3 4 Score: 186

Solution: 5 2 3 4 Score: 186

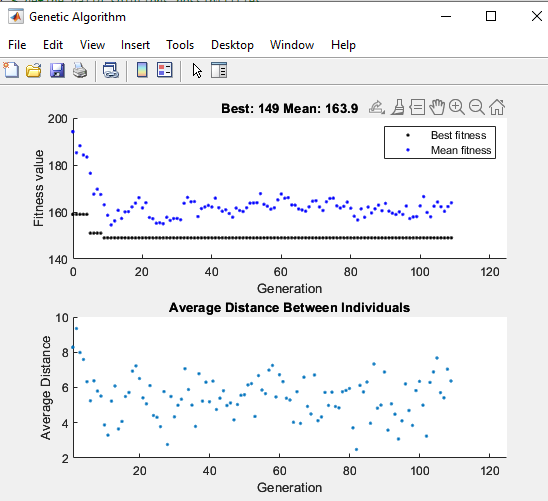
Solution: 1 6 7 12 Score: 173

**Solution: 1 2 7 4 Score: 159**

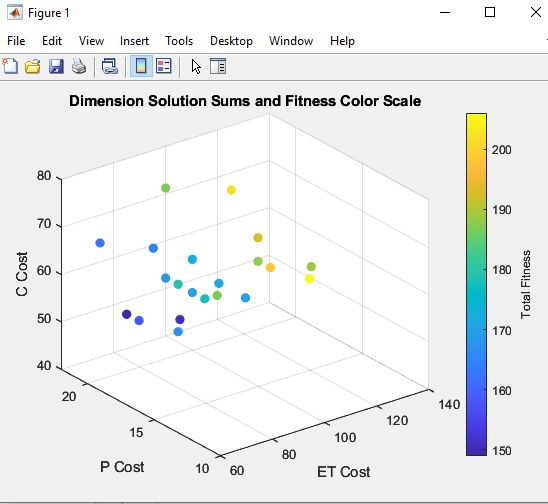
Given that the new Cost dimension moderately impacts the OCF, the GA and myself, tend to stay away from Block 3 bindings. However for the optimal solution the block 3 (P = 2) task 3 (ET = 5) binding has a very low ET and is desirable.  
  
The solution priority weights, for my inspection, are as follows: ET > C > P.

Part B: Use MATLAB to create a visualization for your results in Part A. You can be as creative as you would like to be. At a minimum, show fitness values for each solution in a multidimensional plot. This may be of interest to you:

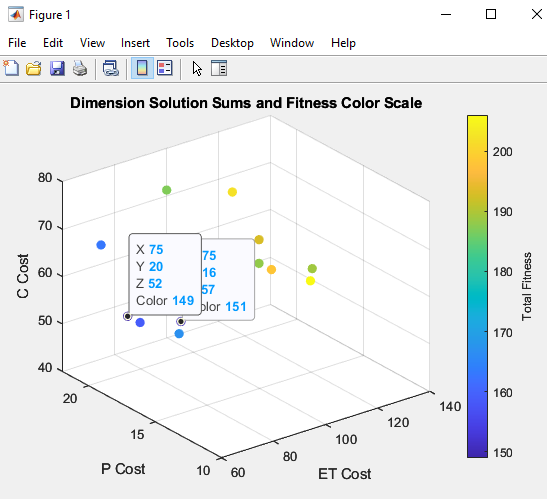
I’ve increased the population size to 150 for the graph to better visualize the outputs.



Here is a 4d graph showing the relationship between the cost sum of each binding, (ET, P, C) and the total fitness including constraints. The color bar helps visualize the total fitness in respect to the other costs.



As we can see, the ET cost is the largest contributing factor for high fitness. Even though we’ve added a Cost (C) to using a specific block, it is still optimal, in some cases, to select a block binding to lower the ET cost. As seen in the below graph with optimal solutions selected.



I’ll admit I couldn’t, easily, find a way to also display the solution values as seen in the GA. Another note is that many of the solutions have overlapping fitness scores, as there were duplicates in the population afterwards.

A very fun little side expedition back into the world of GAs though!