“If you have a question, call Prof Lamont at 937 767-1332”

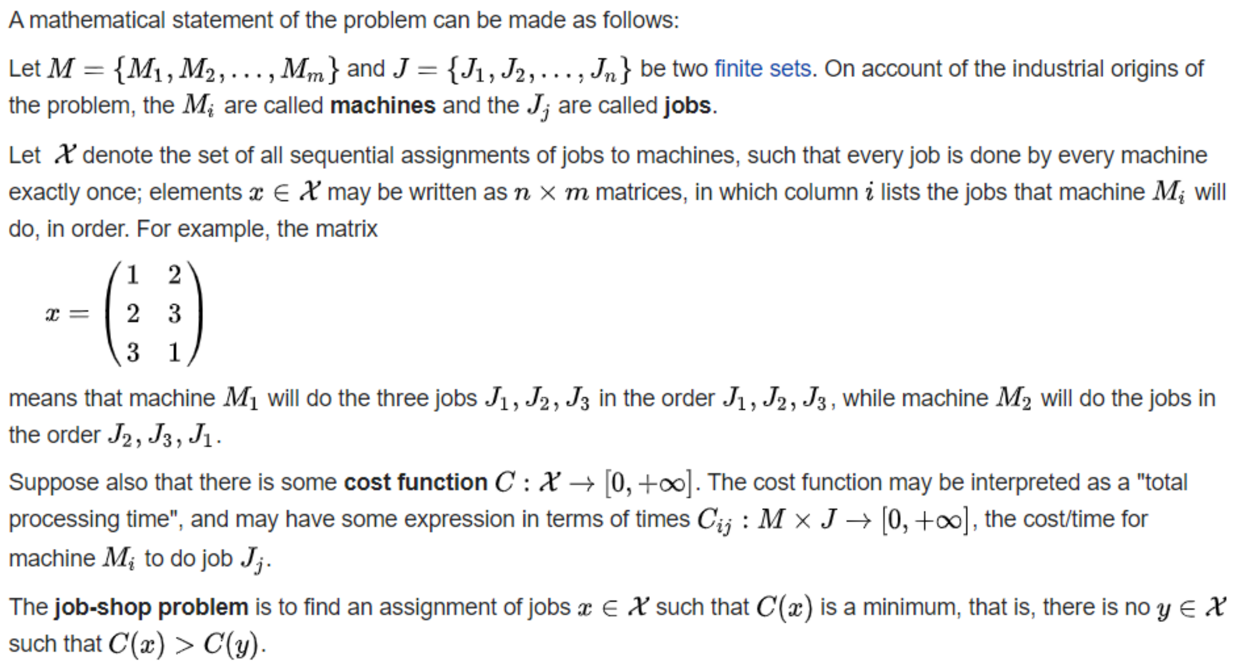
name: John Beighle

Air Force Institute of Technology
Graduate School of Engineering and Management
Department of Electrical and Computer Engineering
CSCE686 Advanced Algorithm Design
Spring Quarter, 2020
Take-at-home Exam; Open book, Internet, …
Indicate References
150 points

1. (10 points) **Where** are *heuristics* brought into the gs_dfs/ bt top-down algorithm design process and for **what** purpose?
    a. **Where?** The example given in MIS_gs_dfsbt_20-2.docx began to introduce heuristics in the “Algorithm Domain Design Specification Refinement” step of the process.
    b. **What purpose?** When finding an optimal solution is impossible or impractical, heuristic (thought of as shortcut or approximation) methods are used to speed up the process of finding a satisfactory solution. A quote from Talbi: “Heuristics can be mental shortcuts that ease the cognitive load of making a decision. Unlike exact methods, [they] allow [us] to tackle large-size problem instances by delivering satisfactory solutions in a reasonable time. There is no guarantee to find global optimal solutions or even bounded solutions.”

2. (30 points) **Job shop scheduling** (JSSP) is an optimization problem in which ideal jobs are assigned to resources at particular times. The most basic version is as follows: We are given *n* jobs $J_1, J_2, ..., J_n$ of varying sizes, which need to be scheduled on *m* identical machines, while trying to minimize the makespan. The makespan is the total length of the schedule (that is, when all the jobs have finished processing). a) **Is** it NPC? b) **Define** as a math/symbolic linear model. c) **Discuss** the applicability and appropriateness of a GA approach. **Present** a selected solution representation along with the resulting GA algorithmic structure including algorithmic operations and parameter values. (Don’t forget standard search elements in algorithm)

    a. **Is it NPC?** No, it is not. It is NP-hard like TSP.
    b. **Mathematical model:** The below statement is from Wikipedia “Job Shop Scheduling”

c. 

A mathematical statement of the problem can be made as follows:

Let $M = \{M_1, M_2, \dots, M_m\}$ and $J = \{J_1, J_2, \dots, J_n\}$ be two finite sets. On account of the industrial origins of the problem, the M_i are called **machines** and the J_j are called **jobs**.

Let $\mathcal{X}$ denote the set of all sequential assignments of jobs to machines, such that every job is done by every machine exactly once; elements $x \in \mathcal{X}$ may be written as $n \times m$ matrices, in which column i lists the jobs that machine M_i will do, in order. For example, the matrix

$$x = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{pmatrix}$$

means that machine M_1 will do the three jobs J_1, J_2, J_3 in the order J_1, J_2, J_3, while machine M_2 will do the jobs in the order J_2, J_3, J_1.

Suppose also that there is some **cost function** $C : \mathcal{X} \to [0, +\infty]$. The cost function may be interpreted as a "total processing time", and may have some expression in terms of times $C_{ij} : M \times J \to [0, +\infty]$, the cost/time for machine M_i to do job J_j.

The **job-shop problem** is to find an assignment of jobs $x \in \mathcal{X}$ such that $C(x)$ is a minimum, that is, there is no $y \in \mathcal{X}$ such that $C(x) > C(y)$.

d. **Discuss the applicability and appropriateness of a GA approach:** It is directly applicable and appropriate. There are many GA based applications to explore. For the approach present the schedules are constructed using a priority rule in which the priorities are defined by the GA. Schedules are constructed using a procedure that generates parameterized active schedules. After a schedule is obtained a local search heuristic is applied to improve the solution. The approach is tested on a set of standard instances taken from the literature and compared with other approaches. The computation results validate the effectiveness of combined technique.

e. **Present a selected solution with algorithmic structure including operations and parameter values:**

The following is from: A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem, J. Gonçalves, J. Magalhães Mendes, M. Resende

Parameter values:

The procedure used to construct parameterized active schedules is based on a scheduling generation scheme that does time incrementing. For each iteration ***g***, there is a scheduling time $\boldsymbol{t_g}$. The active set comprises all operations which are active at $\boldsymbol{t_g}$, i.e. $A_g = \{ j \in J \mid F_j - d_j \leq t_g < F_j \}$. The remaining machine capacity at $\boldsymbol{t_g}$ is given by $RMC_m(t_g) = 1 - \sum_{j \in A_g} r_{j,m}$. $\boldsymbol{S_g}$ comprises all operations which have been scheduled up to iteration ***g***, and $\boldsymbol{F_g}$ comprises the finish times of the operations in $\boldsymbol{S_g}$. Let $\boldsymbol{Delay_g}$ be the delay time associated with iteration ***g***, and let $\boldsymbol{E_g}$ comprise all operations which are precedence feasible in the interval $[t_g, t_g + Delay_g]$, i.e.

$$E_g = \{ j \in J \setminus S_{g-1} \mid \; F_i \leq t_g + Delay_g \;\; (i \in P_j) \}.$$

The makespan of the solution is given by the maximum finish time of all predecessors operations of operation *n+1*, i.e. $F_{n+1} = Max_{l \in P_{n+1}} \{F_l\}$.

The basic idea of parametrized active schedules is incorporated in the selection step of the procedure,

$$j^* = \underset{j \in E_g}{\text{argmax}} \{ PRIORITY_j \}.$$

The set $\boldsymbol{E_g}$ is responsible for forcing the selection to be made only amongst operations which will not cause a delay smaller or equal to the maximum allowed delay. Figure 7 illustrates the selection step.

Pseudocode is represented on the next page:

Pseudocode:

```
Initialization: $g = 1,\ t_1 = 0,\ A_0 = \{0\},\ \mathcal{F}_0 = \{0\},\ S_0 = \{0\},\ RD_m(0) = 1\ \ (m \in M)$

while $|S_g| < n + 1$ repeat
{
   Update $E_g$
    while $E_g \neq \{\}$ repeat
    {
          Select operation with highest priority
          $j^* = \operatorname{argmax}_{j \in E_g} \{ PRIORITY_j \}$
          Calculate earliest finish time (in terms of precedence only)
          $EF_{j^*} = \max_{i \in P_j} \{F_i\} + d_{j^*}$
          Calculate the earliest finish time (in terms of precedence and capacity)
          $F_{j^*} = \min\{ t \in [EF_{j^*} - d_{j^*}, \infty] \cap \mathcal{F}_g \mid r_{j^*,m} \le RMC_m(\tau),$
                     $r_{j^*,m} > 0,\ \tau \in [t, t + d_{j^*}] \} + d_{j^*}$
          Update $S_g = S_{g-1} \cup \{ j^* \}$ , $\mathcal{F}_g = \mathcal{F}_{g-1} \cup \{ F_{j^*} \}$
          Iteration increment: $g = g+1$
          Update $A_g, E_g, RD_m(t) \mid t \in [F_{j^*} - d_{j^*}, F_{j^*}], m \in M \mid r_{j^*,m} > 0$
    }
    Determine the time associated with iteration $g$
    $t_g = \min \{ t \in \mathcal{F}_{g-1} \mid t > t_{g-1} \}$
}
```

3. (10 points) **Why** would multi-crossover points be a problem in GA execution? **Relate** to GA Schema Theorem and epistasis.
   a. Schema theorem equation has exponential growth characteristics as explained on slide 122 of GA Lecture slides. Exponential growth causes serious issues in optimization efforts.

4. (10 points) **How** would you keep population intensification and diversification in a GA population?
   a. Intensification in intermediate memory
   b. Diversification in long term memory

5. (15 points) Given a partition problem, which is to partition a multiset *S* into two subsets *S*$_1$, *S*$_2$ such that the difference between the sum of elements in *S*$_1$ and the sum of elements in *S*$_2$ is minimized (NP-Hard?); **Define** a search algorithm (functional pseudo code) to solve this problem with standard search elements inserted.

NP-Hard – YES! The optimization version of the PP is NP-hard (Wikipedia: Partition Problem)

Continued…

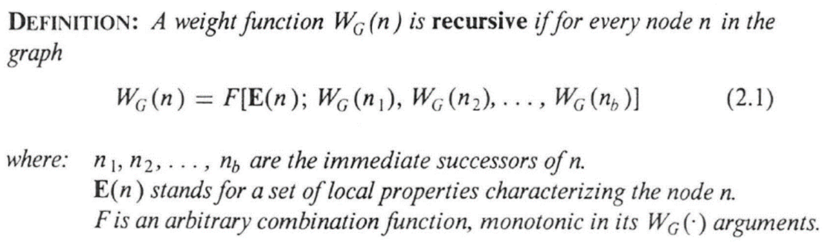
```
The problem is solved easily with recursion. This algorithm could be adjust to output 2 sets.
Di = set of integers {1..n}
Do = 1
--------------------------------------------------------------------
Algorithm FindMin
--------------------------------------------------------------------
While I != 0 Do
   findMinRec(int arr[], int i, int sumCalculated, int sumTotal) do
      // If we have reached last element. Sum of one subset is sumCalculated,
      // sum of other subset is sumTotal- sumCalculated.  Return absolute
      // difference of two sums.
      if (i == 0)
         return Math.abs((sumTotal-sumCalculated) -  sumCalculated);

      // For every item arr[i], we have two choices (1) We do not include it first set
      // (2) We include it in first set We return minimum of two choices
      return Math.min(findMinRec(arr, i - 1, sumCalculated + arr[i-1], sumTotal),
                         findMinRec(arr, i-1, sumCalculated, sumTotal));
 End While
```

6. (10 points) **What** is a matroid problem model? **Why** is it important? **Give** an example .
   a. **What is it?** It constitutes a concept more general than that of just graphs; however, it is a structure that abstracts and generalizes the notion of linear independence in vector spaces.
   b. **Why is it important?** It provides advantages in that it allows two different phenomena to be described in a more uniform way. One of these is that the same matroid corresponds to the weakly isomorphic graphs. The other is, among matroids, the dual always exists. If some graph has no dual, the consequence is merely that the dual of the corresponding matroid cannot be obtained as the matroid of the graph.
   c. **An example:** Finding a minimum cost basis in a partitioning matroid.

7. (5 points) If the Boolean matrix for the SCP is total unimodular, **what** is the impact on the search process?
   a. Can be solved optimally with LP

8. (10 points) **What** combinations of deterministic and stochastic algorithms could be useful in solving NPC problems?
   a. We combined Tabu with deterministic in class homework to deal with local optima. The same can be done with pareto front and linear search methods. These combinations are very useful to approach the NPC problem space.
   b. The GA presented in problem two is also an example of a combination of the two that worked together, A GA to find a solution and linear search to optimize, that demonstrated NPC problem solving applicability.

9. (10 points) **What** is and **why** is the recursive weight calculation important in a best-first search? **Give** example.
   a. **What is it?** See definition below.



DEFINITION: *A weight function $W_G(n)$ is* **recursive** *if for every node n in the graph*

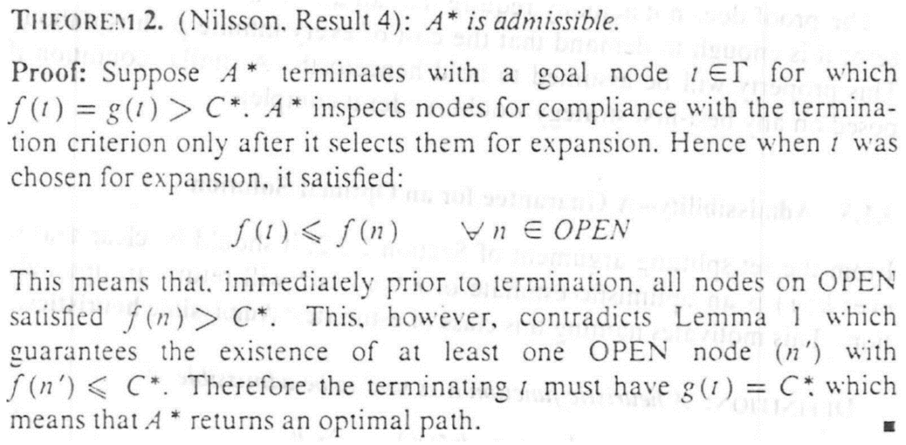
$$W_G(n) = F[\mathbf{E}(n);\, W_G(n_1), W_G(n_2), \ldots, W_G(n_b)] \qquad (2.1)$$

where: *$n_1, n_2, \ldots, n_b$ are the immediate successors of n.*
$\mathbf{E}(n)$ stands for a set of local properties characterizing the node n.
F is an arbitrary combination function, monotonic in its $W_G(\cdot)$ arguments.

b.
c. **Why is it important?** Where the combining function *F* exists it makes it possible to evaluate the merit of a given solution graph from the bottom up, starting at the payoffs associated with the terminal nodes and working upward until the merit of the entire solution graph is computed at the root node.
d. **An Example Problem that could be solved bfs:** The Counterfeit Coin problem with the maximum number of tests as a weight measure

10. (10 points) **What** is and **why** is the ordering-preserving important in a best-first search? **Give** example.

a. What is order-preserving? It is keeping track of the order in which node costs are evaluated in bfs.
b. **Why is it important?** Pearl explains that order-preserving is vital to admissibility proofs e.g. Theorem 2 at the bottom of pg 78 in chap 3.



THEOREM 2. (Nilsson, Result 4): *A* is admissible.*

Proof: Suppose A^* terminates with a goal node $t \in \Gamma$ for which $f(t) = g(t) > C^*$. A^* inspects nodes for compliance with the termination criterion only after it selects them for expansion. Hence when t was chosen for expansion, it satisfied:

$$f(t) \leqslant f(n) \qquad \forall\, n \in OPEN$$

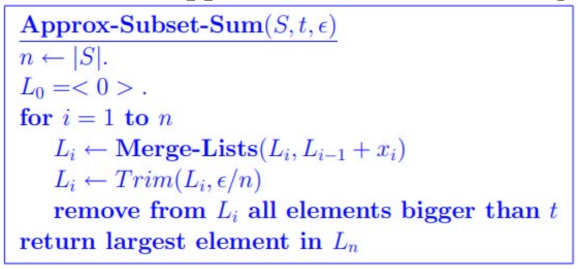
This means that, immediately prior to termination, all nodes on OPEN satisfied $f(n) > C^*$. This, however, contradicts Lemma 1 which guarantees the existence of at least one OPEN node (n') with $f(n') \leqslant C^*$. Therefore the terminating t must have $g(t) = C^*$ which means that A^* returns an optimal path. ■

c.

11. (20 points) **What** makes a NPC problem strong or weak? **Relate** to PTAS and FPTAS approximation possibilities? **Give** two NPC examples of each including approximation techniques.

a. **What makes a NPC problem strong or weak?** According to our Lecture 1 definition: "A problem is said to be strongly NP-complete (NP-complete in the strong sense), if it remains so even when all of its numerical parameters are bounded by a polynomial in the length of the input.
b. **Relate to PTAS and FPTAS approx. possibilities:** From a theoretical perspective any strongly NP-hard optimization problem with a polynomial bounded objective function cannot have a fully polynomial-time approximation scheme (FPTAS – *Talbi p22*)."
c. **Examples of Strongly NPC**: Clique (MIS), Set-Covering (SCP), Bin Packing, 3-Partition, Satisfiability (SAT), Hamiltonian Cycle, TSP, 3-Colorability
   i. **Bin Packing approximation**: Approximation first fit descending (FFD) heuristic
   ii. **MIS approximation:** There is an approx. preserving reduction from MIS to instances of PIPs with $\Delta_1 \leq 2$.
d. **Examples of Weakly NPC**: 0-1 Knapsack, Subset Sum, Partition
   i. **Dual Bin Packing approximation**: First Fit increasing (FFI)

ii. **Subset Sum approximation**: See below algorithm



```
Approx-Subset-Sum($S, t, \epsilon$)
$n \leftarrow |S|$.
$L_0 =< 0 >$ .
for $i = 1$ to $n$
    $L_i \leftarrow$ Merge-Lists($L_i, L_{i-1} + x_i$)
    $L_i \leftarrow Trim(L_i, \epsilon/n)$
    remove from $L_i$ all elements bigger than $t$
return largest element in $L_n$
```

iii.

12. (10 points) **Why** is the TSP NP-Hard and not NPC?

a. Because it takes exponential time to solve NP, the solution cannot be checked in polynomial time. Thus, it is NP-hard, but *not* in NP.