CSE 701

Project 1 & 2

Rana Shariat

Student number: 400355117

Project 1

Context-Free Language

A context-free grammar (CFG) is defined as a tuple $G =< N, S, \Sigma, P >$ such that N is called the set of non-terminals and the elements are in capital letters, S is the starting non-terminal, Σ is the set of terminals and the elements are in lower case letters, and P shows the productions of type $A \to w$ where w is a sequence of terminals and non-terminals and $A \in N$. A parsing tree is tree which have terminal leaves, non-terminal inner nodes, and root S. Also, children of each inner node form w such that $A \to w \in$ grammar. The sequence produced by the left to right leaf nodes of a parsing tree is a sequence recognized by the grammar. The set of all sequences recognized by a grammar is defined as context-free language. A production $A \to w|v$ means either $A \to w$ or $A \to v$. A grammar is in Chomsky normal form if the productions has types $A \to BC$ or $A \to a$.

Example

The following CFG produces sequences which have undetermined number of as followed by undetermined number of bs.

$$S \to AB, \quad A \to AA|a, \quad B \to BB|b$$

For example, sequence *aabbb* could be produced taking the following steps.

$$S \rightarrow AB \qquad \text{(replace A by AA)} \qquad (1)$$

$$AB \rightarrow AAB \qquad \text{(replace B by BB)} \qquad (2)$$

$$AAB \rightarrow AABB \qquad \text{(replace B by BB)} \qquad (3)$$

$$AABB \rightarrow AABBB \qquad \text{(replace A by a)} \qquad (4)$$

$$AABBB \rightarrow AaBBB \qquad \text{(replace A by a)} \qquad (5)$$

$$AaBBB \rightarrow aaBBB \qquad \text{(replace A by a)} \qquad (6)$$

$$aaBBB \rightarrow aabBB \qquad \text{(replace B by b)} \qquad (7)$$

$$aabBB \rightarrow aabbB \qquad \text{(replace B by b)} \qquad (8)$$

$$aabbB \rightarrow aabbb \qquad \text{(replace B by b)} \qquad (9)$$

As mentioned before, each sequence in a grammar is produced by the leaf nodes of a parsing tree in order from left to right. Quimper & Rousseau (2010) developed **Algorithm 1** based on CYK parser algorithm (an algorithm to decide whether a sequence belongs to a grammar or not) which makes a directed acyclic graph (DAG) which embeds all the sequences produced by the CFG called grammar graph. In this graph the leafs show the terminals and their position in the sequences, and each inner node $N(A, i, j)$ is related to non-terminal A starting in position i with span j.

Example

Consider the previous example CGF. Now, we want to make the grammar graph using **Algorithm 1** for sequences of length 3 ($T = 3$). By performing lines 1 to 4, **table 1** would be the set of nodes and their children.

Next, going through line 5 to 12 nodes in **table 2** will be added to **table 1**.

Finally, in line 18 we delete all nodes other than the descendants of $N(S, 1, 3)$. **Fig-**

Algorithm 1: An algorithm for producing the grammar graph

```
for all non-terminals A do
    for i ∈ [1, T] do
        Create node N(A, i, 1)
        Children(N(A, i, 1)) ← {t|A → t ∈ G}
for j ∈ [2, T] do
    for i ∈ [1, T − j + 1] do
        for all non-terminal A do
            Create node N(A, i, j)
            Children(N(A, i, j)) ← {< N(B, i, k), N(C, i + k, j − k) > |
            k ∈ [1, j), A → BC ∈ G
            children(N(B,i,k)) ≠ ∅,
            children(N(C, i+k, j-k)) ≠ ∅}
Delete any node which is not a descendant of node N(S, 1, T).
```

Node	**Children**
$N(S,1,1)$	
$N(S,2,1)$	
$N(S,3,1)$	
$N(A,1,1)$	a
$N(A,2,1)$	a
$N(A,3,1)$	a
$N(B,1,1)$	b
$N(B,2,1)$	b
$N(B,3,1)$	b

Table 1: Possible leaf nodes and their children.

ure 1 shows the resulted grammar graph. Starting from $N(S,1,3)$ one can choose the right or left edge. The sequences will be *abb* or *aab* respectively.

Node	**Children**
$N(S,1,2)$	$< N(A,1,1), N(B,2,1) >$
$N(A,1,2)$	$< N(A,1,1), N(A,2,1) >$
$N(B,1,2)$	$< N(B,1,1), N(B,2,1) >$
$N(S,2,2)$	$< N(A,2,1), N(B,3,1) >$
$N(A,2,2)$	$< N(A,2,1), N(A,3,1) >$
$N(B,2,2)$	$< N(B,2,1), N(B,3,1) >$
$N(S,1,3)$	$< N(A,1,1), N(B,2,2) >, < N(A,1,2), N(B,3,1) >$
$N(A,1,3)$	$< N(A,1,1), N(A,2,2) >, < N(A,1,2), N(A,3,1) >$
$N(B,1,3)$	$< N(B,1,1), N(B,2,2) >, < N(B,1,2), N(B,3,1) >$

Table 2: Possible inner nodes and their children.

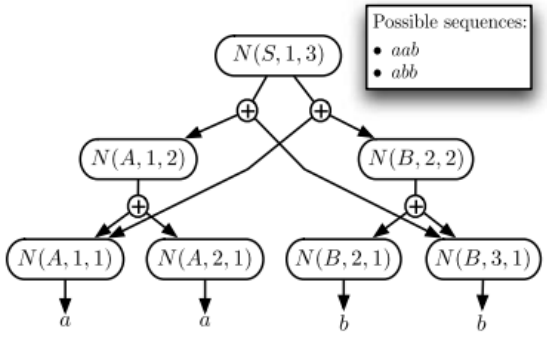

Figure 1: Grammar graph for example grammar with T=3.

In the next step, we need to design an algorithm that chooses the best sequence based on our objective out of this grammar graph. This algorithm is developed based on the algorithm designed by Quimper & Rousseau (2010). **Algorithm 2** starts from the leaf nodes by assigning cost to them. Then it moves through inner nodes, and assigns the minimum sum of costs for each pair of children of each node to that node. Finally, it gives the minimum cost for the root node. Also, the sequence related to this minimum cost can be gained by backtracking in the grammar graph to the nodes with minimum cost in lines 2 and 4. The definition of cost function depends on the definition of problem.

Algorithm 2: An algorithm for choosing the optimal sequence

for *every leaf node* $N(A, i, 1)$ **do**
  $weight(N(A, i, 1)) \leftarrow min\{cost(a, i)|A \rightarrow a \in G\}$
for *every inner node* $N(A, i, j)$ *in post order* **do**
  $Weight(N(A, i, j)) \leftarrow min\{weight(N(B, x, y)) + weight(N(C, w, z))| < N(B, x, y), N(C, w, z) > \in Children(N(A, i, j))\}$
Return weight$(N(S, 1, T))$.

Scheduling Problem

In this project, we want to use CFG for modeling staff scheduling problem. Assume we want to get the optimal schedule for a staff for one working day. Our objective is to minimize the sum of overstaffing and understaffing for all the periods. This staff can be on rest (R, r) which means they have not started the working day or has already finished it. Also, they can be busy by doing activity (A, a). Also, a one period break must be assigned to this staff between the working hours. Using CFG the problem can be formulated as following. ϵ is a null terminal.

$$S \rightarrow RWR, \quad W \rightarrow AbA, \quad A \rightarrow AA|a, \quad R \rightarrow RR|r|\epsilon$$

To use the defined algorithms in the previous part, first we need to change this formulation to Chomsky normal form. The following shows the result.

$$S \rightarrow RT|WR|RW|XA, \quad T \rightarrow WR, \quad A \rightarrow AA|a, W \rightarrow XA, \quad X \rightarrow AY, \quad Y \rightarrow b, \quad R \rightarrow RR|r$$

We use **Algorithm 1** to produce the grammar graph. Next, we need to define a cost function to use **Algorithm 2**. We see that in our problem every non-terminal can yield at most one terminal. Since we just have one staff we define the cost function for period i as following. The cost function shows that if there is $demand(i)$ in period i, and we assign the staff to do activity in that period, the overstaffing or understaffing for that period could be calculated by $|demand(i) - 1|$.

C programming language is used to code the defined algorithms for solving different problems. The inputs of this code are two files, demand.csv and Grammar.txt. The

Algorithm 3: Cost function

```
if a ∈ activity then
    cost(a,i) = |demand(i) − 1|
else
    cost(a,i) = demand(i)
```

first file includes number of staff needed for activity terminal, which is 'a' for the defined problem, for each period. Grammar.txt has 5 lines. The line show activity terminal, non-terminals, terminals, rules related to non-terminals, and rules related to terminals respectively. In line 2 and 3 one just needs to write the non-terminals and terminals without any spaces. In line 4, each 3 letters shows a non-terminal rule. In line 5, each 2 letters show a terminal rule.

Grammarn.txt

```
a
SAB
ab
SABAAABBB
AaBb
```

Table 4 shows the results for 3 different instances for the defined grammar.

Period	1	2	3	4	5	6	7	8	9	10	11	12	Optimal Cost	Optimal Schedule
Instance 1	3	2	0	1	2	1	1	0	0	0	1	0	5	aabaaaarrrrr
Instance 2	3	0	0	1	2	1	0	0	1	1	1	0	5	rrraaabaaaar
Instance 3	3	0	0	1	2	1	0	0	2				6	abaaaarrr

Table 3: Optimal schedules for the given demands.

Another grammar that we will see the results for is the grammar of the example mentioned before. Assume that each employee should start the doing activity (a), and when stops doing it (b) will not be able to continue. Ans, we want to minimize sum of

overstaffing and understaffing. We will change Grammar.txt to the following.

Grammarn.txt

```
a
SAB
ab
SABAAABBB
AaBb
```

Table 4 shows the results for 3 different instances for the defined grammar.

Period	1	2	3	4	5	6	7	8	9	10	11	12	Optimal Cost	Optimal Schedule
Instance 1	3	2	0	1	2	1	1	0	0	0	1	0	6	aaaaaaabbbbb
Instance 2	3	0	0	1	2	1	0	0	1	1	1	0	7	aaaaaaabbbbb
Instance 3	3	0	0	1	2	1	0	0	2				7	aaaaaabbb

Table 4: Optimal schedules for the given demands.

One could simply define new problems and give the Grammar as an input to the C code. Also, a new cost function appropriate for that problem must be defined. Using CFG for optimizing different problems could be much helpful when the problem size increases.

Project 2

In project 1, CFG was implemented to solve general scheduling problems for one employee by minimizing overall over-coverage and under-coverage. In this project, we will solve more complex scheduling problems. We want to solve scheduling problems with multiple activities for multiple employees. Also, some constraints will be implied on the length of the subsequences of our schedules.

1 Enhancing context-free language

Adding constraints on the length of the subsequences could be done by enhancing context-free language by setting valid spans for the subsequence. For example, $A_{S_a} \rightarrow B_{S_b}C_{S_c}$ means the span of literal A, B, and C should be S_a, S_b, and S_c respectively. To implement we need to make a change to Algorithm 1, and add $j \in S_a$, $k \in S_b$, and $j - k \in S_c$ to line 12.

2 Problem Definition

In this problem we assume all employees have the same skills. So, one graph will be constructed for all employees. The number of periods T is larger than 14. During each time slot each employee can rest (r), be on a break (b) or do an activity (t_i).
The following rules are considered:

1. If an employee is working on this day, the working duration should be between 8 and 15 periods.
2. The break, which is one period, must be between periods 6 and 13 after the start of work, .
3. There is a minimum, $min(t)$ and maximum, $max(t)$, duration limit for each activity t_i.

The context free grammar for this problem is as the following.

$S \implies RW_{[8,15]}R|R$

$W \implies A_{[6,13]}bA$

$R \implies Rr \mid r \mid \epsilon$

$A \implies T_tA'_t \mid T_t \quad \forall t$

$T_{t[min(t),max(t)]} \implies T'_t \quad \forall t$

$T'_t \implies tT'_t \mid t \quad \forall t$

$A'_t \implies T_{t'}A'_{t'} \mid T_{t'} \quad \forall t, \forall t' \{t\}$

After changing it to Chomsky normal form, we can generate the grammar graph.

$$S \implies WR|RW|RZ|RR \tag{1}$$

$$Z \implies WR \tag{2}$$

$$W_{[8,15]} \implies A_{[6,13]}Y \tag{3}$$

$$Y \implies XA \tag{4}$$

$$X \implies b \tag{5}$$

$$R \implies RR|r \tag{6}$$

$$A \implies T'_{t[min(t),max(t)]}A'_t|T'_{t[min(t)-1,max(t)-1]}A'_t \qquad \forall t \tag{7}$$

$$A \implies t \qquad \forall t \in \{t|min(t) = 1\} \tag{8}$$

$$T'_t \implies T'_tT'_t|t \qquad \forall t \tag{9}$$

$$A'_t \implies T'_{t'[min(t'),max(t')]}A'_{t'}|T'_{t'[min(t')-1,max(t')-1]}T'_{t'[1,1]} \qquad \forall t' \neq t \tag{10}$$

$$A'_t \implies t' \qquad \forall t' \neq t, \quad t' \in \{t'|min(t') = 1\} \tag{11}$$

3 Cost function

For calculating weight of the leaf nodes, we will use 2. However, since we have multiple activities and staff in this problem, cost function will be defined in a different way. Based

on Algorithm 4, $cost(a, i)$ basically means the overall over-coverage and under-coverage in period i based on the demand and assigned schedules if we assign the current employee to a.

Algorithm 4: Cost function cost(a,i)

```
1 Assume an employee is assigned to 'a' in period i
2 for j ∈ activity do
3     Find the number of employees assigned to activity j in period i
4 Calculate the overall over-coverage and under-coverage for all activities in
    period i based on the previously assigned schedules and assumption of adding a
    new employee.
```

4 Large neighborhood search

Since we have multiple employees, we will use large neighborhood search method to find the local optimal solution for a set of employees. Algorithm 5 shows this method. Residual demand means the unanswered demand, and can be calculated based on the set of schedules for all employees.

Algorithm 5: Large neighborhood search

```
1 while The overall cost is decreased compared to the past iteration do
2     for each employee do
3         Remove the assigned schedule
4         Find the best schedule for the employee based on the cost calculated by
            considering other schedules assigned to the other employees
5         Calculate the overall over coverage and under coverage for all periods
            based on the new set of schedules
```

5 Inputs of the program

The first input file is Grammar.txt. The first line are the non-terminals, the second line terminals, the third line are the non-terminal rules and each 3 character shows one rule. Line 4 shows the terminal rules. Only the rules related to the 6 first lines of the Chomsky

normal form are added to this file and the rules related to tasks will be added by the program. The reason is that when the number of tasks increases it would be tedious to add all the rules manually. The last line shows the number of employees in our problem.

Grammar.txt

```
S,W,R,Z,A,Y,X
b,r
S,W,R,S,R,W,S,R,Z,S,R,R,Z,W,R,W,A,Y,Y,X,A,R,R,R
X,b,R,r
10
```

Next, we have tasks.csv. The columns show the task ID, minimum duration, and maximum duration limit form left to right.

tasks.csv

```
1,1,2
2,2,4
3,2,3
4,3,5
```

The demand.csv includes the demand for each task in each period. Each row and column is related to a task and a period respectively. Each row shows the task having the ID equal to the task in the same row of tasks.csv.

demand.csv

```
3,1,2,2,1,3,2,1,2,0,1,1,1,2,2,1
1,2,2,1,2,1,3,2,1,3,2,2,4,1,0,1
```

```
1,3,2,3,1,4,3,1,1,3,2,1,2,1,5,0
3,1,1,3,2,1,2,1,2,1,3,2,1,1,2,1
```

The last file is the constraints.csv which includes the spanning constraints for non-terminal rules. The first column shows the rule number and the next lines show the spanning limits on each non-terminal in our rule. Again, just the rules related to the first 6 lines of the Chomsky normal form should be added and the ones related to tasks will be added by the program. For example, the following file shows the constraints on the rule on line 3 which is the 6th rule. $1, T$ shows there are no constraints on the last non-terminal which is Y.

constraints.csv

```
6,8,15,6,13,1,T
```

6 Computational result

Tow different instances for the Chosmky normal form defined in section 2 are presented in this section.

Instance 1: Table 5 shows the demand and duration limit for each task. Table 6 shows the local optimal schedules for 6 employees with total cost 17.

Period	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	min	max
Task 1	3	1	2	2	2	3	2	1	2	0	1	1	1	5	2	1	1	2
Task 2	1	3	2	1	2	1	5	2	5	3	2	3	4	4	5	1	2	4

Table 5: Task, demand, and duration limits

Instance 2: Table 7 shows the demand and duration limit for each task for the next instance. Table 8 shows the local optimal schedules for 5 employees with total cost 24.

The constraints on the subsequence lengths could be easily changed using file constraints.csv.

Employee	Schedule
1	1 2 2 1 2 2 2 1 2 2 b 2 2 1 1 r
2	1 2 2 2 1 1 2 2 2 2 b 1 2 2 1 r
3	2 2 1 2 2 1 1 b 1 2 2 1 1 2 2 r
4	r r r r 1 1 2 2 1 1 b 2 2 1 2 2
5	r r 1 1 2 2 1 2 2 b 1 2 2 2 2 1
6	r r r r r r 2 2 2 1 2 2 b 2 2 r

Table 6: Best schedules

Period	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	min	max
Task1	3	1	2	2	1	3	2	1	2	0	1	1	1	2	2	1	1	2
Task 2	1	2	2	1	2	1	3	2	1	3	2	2	4	1	0	1	2	4
Task 3	1	3	2	3	1	4	3	1	1	3	2	1	2	1	5	0	2	3

Table 7: Task, demand, and duration limits

Employee	Schedule
1	1 2 2 1 2 2 b 1 2 2 1 2 2 1 1 r
2	1 2 2 1 3 3 b 3 3 2 2 1 2 2 1 r
3	1 3 3 2 2 1 1 b 1 2 2 2 1 3 3 r
4	r 1 3 3 1 1 2 2 1 b 3 3 3 1 2 2
5	3 3 1 3 3 1 2 2 b 3 3 r r r r r

Table 8: Best schedules

References

Quimper, C.-G., & Rousseau, L.-M. (2010). A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, *16*(3), 373–392.