Production Planning by Shuffle Operation

ADALBERT GOLOMETY, ALINA PITIC, IULIA GOLOMETY, ANTONIU PITIC
Department of Computer Science
Hermman Oberth Faculty of Engineering
Lucian Blaga University of Sibiu
4 Emil Cioran Street, 550025 Sibiu
ROMANIA

Abstract: - This paper presents an implementation of shuffle operation in production planning. We present a computational formula for shuffle and some optimizations to reduce the sets of shuffle strings. Our idea is to combine shuffle with parallelism for a planning of production phases.

Key-Words: - shuffle, production phases, production planning, linguistic model, execution time.

1 Linguistic Model of Production Process

By production process we understand the transformation action of resources (material, energy) in final products according of a fabrication recipe.

The model that we will show has the purpose to determine the set of actions strings that represents right evolutions of process with the help of linguistic mechanism.

A production system is defined as a 7-tuple:

Σ = (O, R, M, A, T, I, S)
where
a) O= a finite and non empty set of finite products.
b) R= a finite and non empty set of resources.
c) M= set of materials.
d) A= set of activities (phases) set 1,2,..,q_i that must be executed for the fabrication of each other of $a_i \in O$ products.
e) T= deliver plan
f) I= economical indicators
g) S= set of fabrication processes

Between the phases from A exists a series of precedence established relations, that can be represented through many graphs Γ_i = ({1, 2,.., qi },U_i). These graphs don't have circuits and always q_i is the last activity from the respective graph. We note with U_i the arcs of graph Γ_i . There are one graph Γ_i for each product $a_i \in O$. By linear the graphs Γ_i results Z_i sets for all products. By successive applications of Shuffle operation results the plan set of fabrication:

$$W = Shuf^*\left(\bigcup_{i=1}^{P} Z_i\right)$$

where p= number of products or sets of products.

We note with *Shuf** the transitive closure for *Shuf* defined in 2.1

2 Shuffle Operation

2.1 Definition

For V a vocabulary and two strings $x,y \in V^*$ the shuffle operation is defined:

$$Shuf(x,y)=\{x_1y_1\, x_2y_2 \dots x_py_p : x_i, y_i \in V^*,\ x=x_1x_2\dots x_p,\ y=y_1y_2\dots y_p,\ p\geq 1\} \qquad (1)$$

The note $x_i,y_i \in V^*$ means that x_i and y_i can be any symbols sequences (substrings) from the two strings x and y.

2.2 Computational formula

The Shuffle operation for two characters set x and y of different lengths has two phases: the two strings division in substrings sequences x_i and y_i and then the mixing of all sequences x_i and y_i that result from the first phase.

2.2.1 The string division

Note a string with $s=a_1 a_2 \dots a_n$ and $|s|$=n. We note by $f(i,s)$ the function for first i string symbols:

$$f(1,s) = a_1$$
$$f(2,s) = a_1a_2$$
....
$$f(n,s) = a_1a_2\dots a_n \qquad (2)$$

Note s^i the characters beginning from i position from s string. One results the next substrings :

$$s^1 = s = a_1a_2\dots a_n$$
$$s^2 = a_2\dots.a_n$$
$$s^3 = a_3\dots.a_n$$

....

$s^n = a_n$.

For example if we have $s=|abc|$ string it can be divided in substrings this way:

$|a|b|c|$, $|a|bc|$,$|ab|c|$, $|abc|$.

The first string $|a|b|c|$ is obtained by :

$f(1,s)\ \ f(1,s^2)\ \ f(1,s^3)$ that's equivalent with $f(1,abc)\ \ f(1,bc)\ \ f(1,c)$

The second string $|a|bc|$ is obtained by:

$f(1,s)\ \ f(2,s^2)$ that's equivalent with $f(1,abc)$ $f(2,bc)$

The third string $|ab|c|$ is obtained by :

$f(2,s)\ \ f(1,s^3)$ that's equivalent with $f(2,abc)$ $f(1,c)$

The last string $|abc|$ is:

$f(3,s)$ that's equivalent with $f(3,abc)$

We can make the next notes for the recurrence formula:

$$f^3(s) = \{ f^2(s),\ f(3,s)\, f^0(0) \} \qquad (3)$$

With $f^3(s)$ we have noted the third order of the function f. With $f^0(0)$ we have noted the order 0 of the f function. Function $f^0(0)$ is applied to the empty string and returns the null string, and performs an write in the exit file of the function before it, namely $f(3,s)$.

$$f^2(s)=\{ f^1(s),\ f(2,s)\, f^1(s^3) \} \qquad (4)$$

$$f^1(s)=\{ f^0(s),\ f(1,s)\, f^2(s^2) \} \qquad (5)$$

2.2.2 Return from the mixing of strings

The $f^0(s)$ function returns null string, it is being used just for the returning of recursive function for $f^1(s)$.

Next for $f^1(s^3)$ results :

$$f^1(s^3)= \{ f^0(s^3),\ f(1,s^3)\, f^0(0) \} \text{ for } |s^3| = 1 \qquad (6)$$

For $f^2(s^2)$:

$$f^2(s^2)= \{ f^1(s^2),\ f(2,s^2)\, f^0(0) \} \qquad (7)$$

$$f^1(s^2)= \{ f^0(s^2),\ f(1,s^2)\, f^0(0) \} \text{ for } |s^2| = 2 \qquad (8)$$

Making the replacements for $s=|abc|$ results :

$f^3(abc) = \{ f^2(abc),\ f(3,abc)\, f^0(0) \}$

exit for $f(3,abc)\, f^0(0)= |abc|$

$f^2(abc)=\{ f^1(abc),\ f(2,abc)\, f^1(c) \}$

$f(2,abc)\, f^1(c)= |ab|\ f^1(c)= |ab| \{ f^0(c),\ f(1,c)\ f^0(0) \}=|ab|\ f(1,c)\, f^0(0)= |ab|c|$

exit=$|ab|c|$

$f^1(abc)=\{ f^0(abc), f(1,abc)\, f^2(bc) \}= |a|\ f^2(bc)$

$f^2(bc)=\{ f^1(bc), f(2,bc)\, f^0(0) \}$

$|a|f^2(bc)=|a| \{ f^1(bc), |bc|f^0(0)\}=\{|a|\ f^1(bc), |a|\ |bc|\ f^0(0) \}$

exit =$|a|bc|$

$|a|f^1(bc)= |a|\ \{ f^0(bc),\ f(1,bc)\, f^0(0) \}$

exit= $|a|b|c|$.

2.2.3 General formula of division

In general for $|s| = n$ we have:

$$f^n(s)= \{\ f^{n-1}(s)\ ,\ f(n,s)\, f^0(0) \} \qquad (9)$$

$f^{n-1}(s)= \{\ f^{n-2}(s)\ ,\ f(n-1,s)\ \ f^1(s^n)\ \}$

$f^{n-2}(s)= \{\ f^{n-3}(s)\ ,\ f(n-2,s)\ \ f^2(s^{n-1})\}$

......

......

$f^3(s) = \{ f^2(s),\ f(3,s)\, f^{n-3}(s^{3+1}) \}$

$f^2(s)=\{ f^1(s),\ f(2,s)\, f^{n-2}(s^3) \}$

$f^1(s)=\{ f^0(s),\ f(1,s)\, f^{n-1}(s^2) \}$

For any x between n and 1 we have the general formula :

$$f^x(s) = \{f^{x-1}(s), f(x,s) f^{n-x}(s^{x+1})\} \qquad (10)$$

with $|s| = n$.

2.2.3 Strings mixture formula

For two strings:

$s_1 = a_1 a_2 .... a_n$

$s_2 = b_1 b_2 .... b_n$

and notes:

$$x^0(s_1,s_2) = s_1 s_2 = a_1 a_2 .... a_n b_1 b_2 .... b_m \qquad (10)$$

$x^1(s_1,s_2) = a_1 a_2 .... a_{n-1} b_1 a_n b_2 .... b_m$

$x^2(s_1,s_2) = a_1 a_2 .... a_{n-2} b_1 a_{n-1} b_2 a_n b_3 .... b_m$

...

$x^k(s_1,s_2) = a_1 a_2 .... a_{n-k} b_1 a_{n-k-1} b_2 ..... a_n b_{k+1} b_{k+2} .... b_m$

$k < m$

For k=n-1 on achieve the start of string s_1

$\Rightarrow$ n-k=n-(n-1)=1 and $a_{n-k}=a_1$
k+1=n-1+1=n and $b_{k+1}= b_n$

Later we have the cases:

$x^{n-1}(s_1,s_2)=a_1b_1a_2b_2...a_nb_nb_{n+1}...b_m$ for n<m

$x^{n-1}(s_1,s_2)=a_1b_1a_2b_2....a_nb_n$ for n=m

$x^{n-1}(s_1,s_2)=a_1b_1a_2b_2...a_nb_ma_{n+1}...a_n$ for n>m

....

$x^{n}(s_1,s_2)=b_1a_1b_2a_2...b_{m-n}a_nb_{m-n-1}...b_m$ for n<m

$x^{n}(s_1,s_2)=b_1a_1b_2a_2...b_ma_na_{m+1}....a_n$ for n>m

....

$x^{n+1}(s_1,s_2)=b_1b_2a_1b_3a_2....b_{m-n-1}a_nb_n....b_m$

....

$x^{n+m-1}(s_1,s_2)=b_1b_2...b_ma_1a_2....a_n$

2.2.4 General formula for shuffle

By generalization of the two phases of shuffle formula (10) and (11) we obtain the final formula of shuffle.

For string $s_1=a_1a_2....a_n \overset{\text{shuffle}}{=====>} f^n(s_1)=S_1$

$s_2=b_1b_2...b_m \overset{\text{shuffle}}{=====>} f^m(s_2)=S_2$

We note with

$$X^0(S_1,S_2)=\{x_0(s_i,s_j)|s_i\in S_1,s_j\in S_2\} \quad (12)$$

the mixture of all substrings from S_1 and S_2 obtained by function x_0 . Also we extend the notation to $X^1,X^2,...,X^{n+m-1}$ and we obtain the final formula :

SHUFFLE(s_1,s_2)=
$\{x^0(f^n(s_1),f^m(s_2)),x^1(f^n(s_1),f^m(s_2))...x^{n+m-1}(f^n(s_1),f^m(s_2))\}$

or a shorter notation

SHUFFLE(s_1,s_2)=$\{x^k(f^n(s_1),f^m(s_2))|k=0,1,2,...,n+m-1\}$

The last formula is the computational formula for Shuffle with two strings s_1 and s_2.

3 Shuffle Implementation

The source code in Visual FoxPro for implementation of the function $f^n(s)$ is show below. First we suppose that an element of a string s has one char length:

```
FUNCTION fn
PARAMETERS sir_f,x,p
* sir_f=string created before
* elements of string sir_f are separated by "|"
* x= function order
* p = position in initial string to be process
 PRIVATE sir_ramas,n
  sir_ramas=SUBSTR(sir_ini,p,x1-p+1)
  n=LEN(sir_ramas)  && length of remain
                              string
  IF  x=0.and.LEN(sir_ramas)=0  && f[0](0)
                     exit from recursive function
     SELECT sir_gen && exit file
     APPEND BLANK
     replace string_gen WITH sir_f+"|"
     RETURN ""
   ENDIF
   IF x=0 && f[0](s) = ""
          RETURN ""
    ENDIF
* generation of expression: f[x-1](s)
   =fn(sir_f,x-1,p)
*generation of expression:  f(x,s) f[n-x](s[x+1])
    sir_f2=sir_f+"|"+substr(sir_ini,p,x)      &&=
                    f(x,s) = string created before
    =fn(sir_f2,n-x,p+x)
RETURN
```

For the mixture of two strings with one char per symbol we use the code below.

```
PROCEDURE mixt_2strings
 SELECT 1
 USE sir_shuffle excl
 ZAP
 sir_1="abcd" && test string 1
 sir_2="wuxyz"  && test string 2
 n=LEN(sir_1)
 m=LEN(sir_2)
 FOR i=0 TO n+m-1
  sir_f=""
  n1=n-i
  m1=ABS(n1)+1
  IF n1>0 then
    ** 1 - n1 = sir_1
    sir_f=sir_f+left(sir_1,n1)
    c=MIN(n,m+n1)
```

```
    FOR j=n1+1 TO c
      sir_f=sir_f+SUBSTR(sir_2,j-n1,1)+
SUBSTR(sir_1,j,1)
    ENDFOR
    IF c>=n && remaining sir_2
      sir_f=sir_f+substr(sir_2,c-n1+1,m-(c-n1))
     ELSE && remaining sir_1
   sir_f=sir_f+SUBSTR(sir_1,c+1,m-c)
 ENDIF
  ELSE
    ** 1 - m1 = sir_2
    sir_f=sir_f+left(sir_2,m1)
    c=MIN(m,n+m1)
    FOR j=m1+1 TO c && n1+1-c=sir_2+ sir_1
      sir_f=sir_f+SUBSTR(sir_1,j-m1,1)
                          +SUBSTR(sir_2,j,1)
    ENDFOR
    IF c>=m && remaining  sir_1
        sir_f=sir_f+substr(sir_1,c-m1+1,n-(c-1))
      ELSE && remaining sir_2
        sir_f=sir_f+substr(sir_2,c+1,m-c)
    ENDIF
  ENDIF
  SELECT sir_shuffle
  APPEND BLANK
  replace string_gen WITH sir_f
ENDFOR
RETURN
```

In production planning each symbol of strings is an execution phase of manufacturing process. Each symbol contain information about product (product code), operation to be performed in this phase (operation code) and the manufacturing series

In our implementation each symbol is a 32 characters length.

A product may have many execution phases. The execution phases will be done in a sequential manner. Execution phases of one product make a string for shuffle operation.

We have a number of strings equally with the number of products (for different products) or equally with the numbers of manufacturing sets (for one product manufactured in many sets). Each set have a unique number, a series number.

Each execution phase have a unique execution time and will be done on one type of manufacturing post. Each post type may have one or many work places. Moreover each work place has its own start time for effective execution.

By shuffle operation on all the strings results a set of string of possible configurations of execution phases. For planning we scan the configurations set and append the start time, final time and execution time for each phase. At last we detect the minimum execution time for each configuration.

4 Optimization

We implemented our model first for one product manufactured in many sets**.** Each string of execution phases has completed with the series number of execution set.

4.1 Eliminating duplicate configurations

After shuffle operation we obtained a large number of configurations set.

For a product with only 2 execution phases we obtained the below number of configurations

Sets number	Configurations set
2	12
3	768
4	245760

On a better look of configurations we observed duplicate configuration.

The first step in our optimization was to eliminate all duplicate configurations by a new scanning in configuration set. After we obtained:

Sets number	Configurations set
2	6
3	90
4	2520

4.2 Reducing the configurations number

Even after elimination of duplicate configuration the number of configurations is high.

After the planning step we make a summary of configurations that have the same execution time. For a set of 90 configurations we obtained only 7 different execution times.

Config	Ziua	Minute	Nr_val
TAI_BAGH | INDOI_B | TAI_BAGH | INDOI_B | TAI_BAGH | INDOI_B |	06/23/06	260	6
TAI_BAGH | TAI_BAGH | INDOI_B | INDOI_B | TAI_BAGH | INDOI_B |	06/24/06	30	12
TAI_BAGH | INDOI_B | TAI_BAGH | TAI_BAGH | INDOI_B | INDOI_B |	06/23/06	190	12
TAI_BAGH | TAI_BAGH | INDOI_B | TAI_BAGH | INDOI_B | INDOI_B |	06/23/06	290	24
TAI_BAGH | TAI_BAGH | TAI_BAGH | INDOI_B | INDOI_B | INDOI_B |	06/23/06	40	24
TAI_BAGH | TAI_BAGH | TAI_BAGH | INDOI_B | INDOI_B | INDOI_B |	06/23/06	340	12

Fig.1 Summary confuration 3 sets with 2 phases

In fig.1 the two execution phases are named TAI_BAGH and INDOI_BAGH. There a 3 sets (series) of the same product manufacturing.

By analyzing the configurations that have the same execution time we extract the following rule for reduce the configurations set.

In a real production planning one execution phase may be done on many work places. If exits many work place for the same type of execution phase, these phases for different series will be executed in parallel. So the order of these phases in configurations set may be anyone. That's the rule we have extracted.

With this rule we obtained the reduction below:

Phases	Sets	Configurations	Reduced configurations
2	3	90	6
7	4	3432	9

4.3 Fluent allocation of manufacturing posts

The next steps in planning optimization are related to use more parallelism and fluent distribution of phases to posts.

To increase the use time of work posts is useful to have more sequences with different series interlaced. For two phases and three series 001,002,003 the following sequences have long execution times:

003 002 001 001 002 003
002 001 003 003 001 002

Sequences have the same characteristic, the series are embedded: 001 in 002 in 003 or 003 in 001 in 002.

Sequences like 001 002 003 001 002 003 or 002 001 003 002 001 003 have a minimum execution time. The last series are interlaced and increase the parallelism in distribution of phase to work posts.

We changed the code for the mixture of two strings $s_1 = a_1 a_2 .... a_n$ and $s_2 = b_1 b_2 .... b_n$. The new code returns first

$$x^{n-1}(s_1, s_2) = a_1 b_1 a_2 b_2 \ldots a_n b_n b_{n+1} \ldots b_m$$

and not:

$$x^0(s_1, s_2) = s_1 s_2 = a_1 a_2 .... a_n b_1 b_2 .... b_m$$

In this manner we obtain first configurations with a minimum execution time.

After the last optimization we observed that the same configuration may have different execution times. The difference was in the distribution step. The high execution times are obtained by a distribution of a sequence of two phases of the same series with the same type of manufacturing post to different work places.

The last step in our optimization was to keep the same post assigned to a series how long it is possible. If the next phase of the same series may be executed on the same type of manufacturing post, this phase will be executed on the same post. This fact increases the fluent allocation of manufacturing posts and also the use time of work posts.

5 Conclusion

Modeling production planning by shuffle operations generates a high number of configurations.

Introduction of parallelism in production planning minimizes the execution times.

Our final conclusion is that the shuffle can't replace the parallelism. The linguistic model with shuffle may be replaced by a model with more parallelism.

One must reconsider parallelism between the product parts and between the different series of the same product. In our point of view a model with matrix grammar is a better solution

References:

[1] Jay Gorsher, Shullfe languges, Petri nets and context sensitive grammars, *Comunications of ACM*, vol.24 isue 9 , 1981, pp. 328-342

[2] S. Ginsburg and A. Greibach, Abstract families of language, *American Math. Society Memoirs*, 1969, pp.30-76

[3] Th. Rus, *Mecanisme formale pentru specificarea limbajelo*r, Ed.Bucharest: Academiei, 1983, pp. 52-55, 86-96.

[4] Gh. Paun, *Mecanisme generative ale proceselor economice*, Ed.Bucharest: Tehnica, 1980, pp. 48-53, 112-116

[5] C. Martin-Vide, Formal languages for linguists: classical and nonclasical models, in *2001 Proc. TALN Conf.*, pp. 26-51

[6] A.Golomety, Horizontal parallel grammars, *Proc of 5thRoedunet IEE International Conference,* Sibiu Romania, 2006, pp.300-305

[7] A.Golomety, E.M.Popa, Formal modeling by a bi-parallel grammar, *WEAS Transaction on Information Science and Application*, volume 4 January, 2007, pg.139-144, ISSN:1790-0832