Χρονικός Προγ/σμός Έργων

Ομαδική Εργασία

Σχεδιασμός βάρδιας προσωπικού σε πολλαπλά έργα

Shift planning in multiple projects environment

Σχεδιασμός βάρδιας προσωπικού σε πολλαπλά έργα

- Το πρόβλημα: Μεγάλη εταιρεία ανάπτυξης λογισμικού διαχειρίζεται ένα σύνολο N έργων τα οποία πρέπει να ολοκληρωθούν μέσα σε D ημέρες. Κάθε μέρα διακρίνεται σε S βάρδιες εργασίας. Κάθε έργο *i* (*i*=1,2,…,N) έχει διάρκεια w_i χρονικές περιόδους (1 περίοδος ισοδυναμεί με 8-ωρη ημερήσια βάρδια εργασίας) και απαιτεί α_i εξειδικευμένους εργαζομένους (πολύ έμπειρους προγραμματιστές) για την εκτέλεση του. Το κόστος εργασίας c_k (k=1,2,…,S) ανά βάρδια διαφέρει. Για παράδειγμα, η αμοιβή για νυχτερινή βάρδια εργασίας είναι μεγαλύτερη από την απογευματινή, ενώ η αμοιβή στην πρωινή βάρδια είναι η χαμηλότερη. Με k=1 ορίζεται η πρώτη βάρδια της ημέρας και με k=S η τελευταία. Υπάρχουν εξαρτήσεις τύπου τέλους-αρχής (Finish to Start) μεταξύ των έργων.
- Το ερώτημα: Πόσο προσωπικό να απασχοληθεί σε κάθε βάρδια ώστε να εκτελεστούν όλα τα έργα εντός του χρονικού ορίζοντα των D ημερών με το ελάχιστο κόστος μισθοδοσίας; **minimize $\Sigma c_j * z_j$**

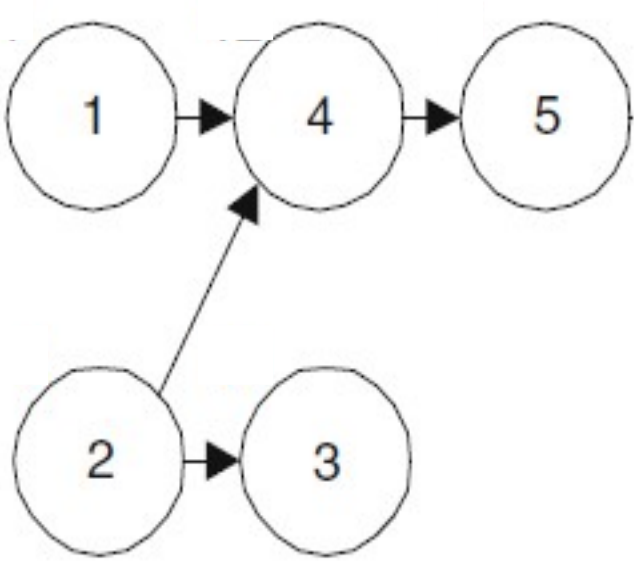
Υποθέσεις στο πρόβλημα

- Τα έργα έχουν εξαρτήσεις με περιορισμούς προήγησης της μορφής τέλους-αρχής (finish-to-start).
- Κάθε έργο είναι διαθέσιμο στον (προ-υπολογισμένο με την CPM) χρόνο νωρίτερης έναρξης (earliest start, ES). Δηλ. μπορεί εν δυνάμει να ξεκινήσει από τον χρόνο ES λαμβάνοντας φυσικά υπόψη τους υφιστάμενους περιορισμούς προήγησης.
- Το προσωπικό είναι ευέλικτο με ταυτόσημες ικανότητες και μπορεί να εργαστεί σε οποιοδήποτε έργο.
- Το προσωπικό εργοδοτείται για να εργαστεί σε μια συγκεκριμένη βάρδια και δεν μπορεί να μετακινηθεί σε άλλη βάρδια.
- Η διάρκεια w_i κάθε έργου καλύπτει έναν ακέραιο αριθμό περιόδων και πρέπει να είναι μικρότερη από τον συνολικό διαθέσιμο χρονικό ορίζοντα, $w_i \leq S*D$.
- Το κόστος αμοιβής c_k για την βάρδια k ικανοποιεί την σχέση: $c_k \geq c_m$ για $k>m$.
- Δεν επιτρέπονται υπερωρίες.

Παράδειγμα

- Έστω τα δεδομένα του πίνακα για N=5 έργα με τις εξαρτήσεις που φαίνονται στο σχήμα.
- Τα έργα πρέπει να ολοκληρωθούν σε **D=5 μέρες** θεωρώντας **S=3 βάρδιες** την ημέρα.
- Επίσης, θεωρείστε για λόγους ευκολίας ότι το κόστος εργασίας κάθε εργαζομένου ανά βάρδια κατανέμεται ως εξής: 1η βάρδια κόστος 1€, 2η βάρδια κόστος 2€ και 3η βάρδια κόστος 3€.

Project *i*	w_i	α_i
1	4	6
2	3	10
3	5	4
4	6	8
5	2	1



Παράδειγμα σχεδιασμού βάρδιας προσωπικού σε πολλαπλά έργα

- Η μορφή της λύσης στο πρόβλημα ανάθεσης:

	βάρδια-1 (ημέρες)										βάρδια-2 (ημέρες)										βάρδια-3 (ημέρες)									
Έργο	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
1																														
2																														
3																														
4																														
5																														

- Αλγόριθμοι επίλυσης:
  - Ο αλγόριθμος της ***μικρότερης αύξησης κόστους*** (**Lowest Cost Increment, LCI**)
  - Ο αλγόριθμος ***έρευνας μεταβλητής γειτονιάς*** (**Variable neighbourhood Search, VNS**)

Algorithm – 1
(*Lowest Cost Increment*)

Step 1:List all periods in the planning horizon.
Step 2: Identify the projects with no predecessors in the network dependency diagram.
Step 3: From the projects identified in Step 2, select that with the ES (earliest start) time. In case of tie select the project with the highest manning requirements. In case of a new tie choose arbitrary.
Step 4: Reserve the earliest available period (taken into account the ES time of the project) which results to the smallest cost increment for the operation of this project. Erase the reserved period from periods list.
Step 5: Repeat from Step 4 until the periods reserved for the operation of the project exactly equals its time duration. Erase the project from the network dependency diagram.
Step 6: Repeat from Step 3 until the network dependency diagram. is empty.

Algorithm 2: The VNS algorithm

VNS = Variable neighbourhood search

VNS: The solution representation

- Let a problem with ***n*** projects.
- A solution **x** in VNS is a vector with ***n*** floating-point numbers taken values in the range (0,1).
- The components of **x** represent the priority of each project.
- For example, for n=5 the vector **x= {0.5, 0.2, 0.1, 0.9, 0.3}** means the following:
  - **Project 1 has priority 0.5, project 2 has priority 0.2, etc.**
- **Projects' operation is assigned to periods using the logic of the LCI algorithm; taken into account ES time and the priorities given in x (not manning requirements).**
- **More details on VNS will be explained in the class.**

Algorithm – 2: VNS algorithm

```
Generate initial solution x
Compute the cost of x
while stopping condition is not met do
    k ← 1
    while k ≤ k_max do
        x' ← Shake(x),
        x'' ← Local Search(x');
        Compute the cost of x''
        if (x'' is better than x)
            x ← x''; k ← 1;
        else
            k ← k+1;
    end-while
end-while
```

```
Shake(x)
{
    Exchange(x);
    Shift(x);
    Exchange(x); Shift(x);
    Shift(x);
    return(x)
}

Local_Search(x')
{
    2-opt(x');
    return(x')
}
```

To compute the cost of x we apply LCI.

The schemes in function *Shake*

Exchange()

Old solution

New solution

Shift()

Old solution

New solution

The 2-opt algorithm

1. Start from the initial solution x
2. Compute the cost c of x
3. for (i=1; i<=n-1; i++)
   for (j=i+1; j<=n; j++)
   {
   Create a new solution x' exchanging the components i and j of x;
   Compute the cost c' of x';
   if (c'< c)
   {
   x=x'; c=c';
   }
   }
4. Stop and return x

To compute the cost of x we apply LCI.