Ottimizzazione della distribuzione del carico di lavoro, in funzione della capacità produttiva assegnata (Load Levelling)

Descrizione dei requisiti – SDG Consulting – Siplus – rif. 2017-10-WAZ Rel. 1.1 01.11.2017

# Descrizione del problema

Il Cliente è una Maison che opera nel mercato del lusso, e produce capi di abbigliamento ed accessori.

Per la produzione, il Cliente si avvale di risorse interne, la cui disponibilità viene messa a disposizione sulla base di un certo numero di ore alla settimana. Per semplificare indicheremo tali risorse con il termine di “fornitore”.

La disponibilità (espressa in termini di ore/settimana) del fornitore a soddisfare le richieste di produzione varia settimanalmente.

Per ciascuna delle merci di cui è richiesta la produzione esiste una valutazione temporale delle risorse necessarie per completare il lavoro.

Dal momento in cui una determinata lavorazione viene richiesta, questa sarà portata a compimento per una data che dipende dal numero di ore stimato per la produzione, e dalla capacità produttiva erogabile dal fornitore. Per esempio, se il tempo di produzione è valutato in 100 ore, e la risorsa che fornisce il lavoro ha una capacità produttiva di 50 ore/settimana, allora il prodotto finito sarà consegnato in due settimane.

Ciascuna produzione assegnata in una data iniziale (espressa in settimana dell’anno), se non può essere completata immediatamente deve essere frazionata e posticipata alle settimane successive, sulle quali però, graverà anche il peso della produzione che nel frattempo sarà stata ad esse attribuita.

Il progetto ha l’obiettivo di sviluppare un algoritmo che permetta di distribuire in maniera ottimale i carichi di lavoro, allo scopo di fornire una previsione sulla consegna del prodotto finito, e di uniformare i carichi di lavoro. Il tutto è mirato ad una simulazione della produzione allo scopo di plannig.

Di settimana in settimana, il programma deve tenere conto delle lavorazioni richieste e delle ore lavorative che il fornitore può erogare per soddisfare tali richieste.

Le produzioni che non possono essere soddisfatte nella settimana attuale, dovranno essere posticipate alle settimane successive, fino al completamento del lavoro. Nel caso in cui in una settimana sia ancora disponibile una capacità lavorativa, sarà possibile anticipare lavorazioni relative alle settimane successive.

# Dati di Input e Output

Il flusso di lavorazione del programma è molto semplice: Il dato di input è un dataset costituito da record che contengono le informazioni relative alle lavorazioni da effettuare, alla disponibilità del/dei produttori, e le relative tempistiche (“Required” sono le ore totali richieste per la lavorazione, “Capacity” è la quantità di ore erogabili dal fornitore per la specifica settimana). Alcuni campi del record sono lasciati vuoti, in essi dovrà trovare posto il risultato della elaborazione. Questi campi sono:

* TCH\_WEEK campo contenente la settimana per cui viene eseguita l’allocazione (non è necessariamente la settimana corrente).
* ALLOCATED è la quantità di ore erogabili nella settimana corrente (“Capacity”) che è stata assegnata a TCH\_WEEK. Potrà essere tutta la “Capacity”, o una sua frazione.
* NOT\_ALLOCATED è la frazione di ore “Required” di TCH\_WEEK che rimane ancora da allocare (se 0, allora l’allocazione di TCH\_WEEK è stata completata).

Il programma prende in carico le informazioni fornite, esegue l’elaborazione richiesta, e compila i relativi campi del dataset.

Lo stesso dataset iniziale, opportunamente compilato, viene prodotto in output.

Di seguito la descrizione dei record che compongono la tabella di input (sono evidenziati in azzurro i campi che entrano in gioco nella implementazione dell’algoritmo di ottimizzazione).

|  |  |
| --- | --- |
| ID | (INT IDENTITY(1,1)) Identificativo univoco del record |
| PRODUCTION\_CATEGORY | (CHAR(2)) Categoria di produzione (vedi relativa tabella) |
| IND\_SEASONAL\_STATUS | (NVARCHAR(128)) Industrial Seasonal Status |
| TCH\_WEEK | (CHAR(6)) In questo campo il programma scrive la settimana in cui viene servita la richiesta relativa a WEEK\_PLAN |
| PLANNING\_LEVEL | (CHAR(10)) Categoria identificativa del tipo di produzione |
| EVENT | (CHAR(10)) Evento (relativo a marketing, vendite ecc…) cui si riferisce il prodotto |
| WEEK\_PLAN | (CHAR(6)) Identificativo della settimana in formato YYYYWW |
| Ahead | (FLOAT) Anticipo. Indica il range temporale, in termini di settimane, secondo il quale a partire dalla settimana attuale, è possibile shiftare le richieste pendenti in avanti verso le settimane successive (se questo valore è zero, la lavorazione della settimana non permette posticipi) il valore float andrà troncato in integer. |
| Late | (FLOAT) Ritardo. Indica quale sia il range temporale, in termini di settimane, secondo il quale, a partire dalla settimana attuale, sia possibile soddisfare le richieste delle settimane precedenti (se questo valore è zero, non posso soddisfare le richieste delle settimane precedenti). Il valore float andrà troncato a integer[[1]](#footnote-1). |
| Priority | (FLOAT) Priorità di esecuzione della richiesta (la priorità 1 è la più elevata). Troncato a integer. |
| Capacity | (FLOAT) Capacità produttiva espressa in ore per la settimana WEEK\_PLAN |
| Required | (FLOAT) Numero di ore necessarie per eseguire la produzione assegnata alla settimana WEEK\_PLAN |
| 1 | (SMALLINT) Ignorare |
| 2 | (SMALLINT) Ignorare |
| 3 | (SMALLINT) Ignorare |
| 4 | (SMALLINT) Ignorare |
| Plan\_BU | (CHAR(10)) Identificativo della produzione (vedi relativa tabella) |
| Flag\_HR | (CHAR(1)) Flag High Rotation. Identifica se si tratta di un prodotto ad alta rotazione, cioè la cui fornitura deve essere ripristinata frequentemente (vedi relativa tabella) |
| ALLOCATED | (FLOAT) Numero di ore allocate per la settimana corrente (frazione della produzione “Required” soddisfatta nella settimana attuale). |
| NOT\_ALLOCATED | (FLOAT) Numero di ore di “Required” non allocate per la settimana corrente (da riportare alle successive settimane, in funzione di “Late”). |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **PRODUCTION\_CATEGORY** | | | 99 Generica | | AB – ABBIGLIAMENTO | | BO – BORSE | | CI – CINTURE | | OG – OGGETTISTICA | | PP - PICCOLA PELLETTERIA | | R1 - RTW TUTE | | R2 - RTW CAMICIE | | R3 - RTW GONNE | | R4 - RTW PANTALONI | | R5 - RTW GIACCHE | | R6 - RTW CAPPOTTI | | R7 - RTW CABAN | | R8 - RTW GIUBBOTTI | | RA - RTW TOP | | RC – BERRETTI | | RD – GUANTI | | RE – SCIARPE | | RG – DENIM | | RH – JERSEY | | RI - ABITI LUNGHI | | RJ - ABITI DA DONNA | | RK - ABITI DA UOMO | | RL – MAGLIERIA | | RM - PELLE & PELLICCE | | RZ - RTW ALTRO | | S0 – SNEAKER | | S1 – TACCO | | S2 – BALLERINA | | S3 – SLIPPER | | S4 – CASUAL | | S5 – FORMALE | | S6 – SANDAL | | S7 – BOOT | | S8 – ESPADRILLES | | SH – CALZATURE | | VA – VALIGIE | | |  | | --- | | **PLAN\_BU** | | | 58\_LG - Leather Goods | | 58\_RTW - Ready To Wear | | 58\_SH – Shoes | | 58\_RTW - Ready To Wear | | |  | | --- | | **FLAG\_HR** | | | N - Not Defined | | F - Flexy | | H - High Rotating | |

Il programma è implementato come libreria di classi scritta in c# (dll), che espone un metodo che viene richiamato dall’applicativo. Negli argomenti di tale metodo vengono passate tutte le informazioni necessarie per eseguire la elaborazione, e cioè:

1. Il dataset completo relativo al database (non è ancora stato deciso se si tratterà di un dataset, o di un file di testo in formato csv, o altro. A scopo di sviluppo utilizzeremo per il momento una tabella appoggiata su SQL Server).
2. Il nome della tabella del dataset
3. L’operazione da eseguire. Per il momento esiste una sola operazione da eseguire, ma il parametro è pensato per usi futuri.

A valle della elaborazione viene restituito il dataset opportunamente compilato con i valori di output.

# Le regole del gioco

Per ciascuna delle settimane contenute nella colonna WEEK\_PLAN bisogna considerare la quantità di lavoro richiesta “Required” (cioè il numero di ore che si rendono necessarie per eseguire la lavorazione).

Le richieste vanno ordinate per priorità “Priority” crescente (perché la priorità più elevata è quella con il valore numerico più basso).

Alla settimana corrente deve essere assegnato il carico di lavoro richiesto, in ordine di “Priority”, in funzione della disponibilità “Capacity” assegnata alla settimana stessa. Se in una settimana sono richieste due lavorazioni con la stessa “Priority”, queste vanno assegnate in ragione della reciproca percentuale.

Se il carico di lavoro richiesto per una determinata lavorazione non può essere soddisfatto dalla “Capacity” erogata nella settimana in analisi, allora la frazione rimanente non assegnata del lavoro (ed eventualmente i lavori non assegnati) devono essere ripartiti nelle settimane successive. La lavorazione potrà essere eseguita al massimo per “Late” settimane successive.

Il meccanismo secondo il quale ciascuna settimana prende in carico il lavoro assegnato, è descritto di seguito, considerando il caso più generale che esclude la prima settimana contenuta nel dataset (per la quale, ovviamente, non esistono operazioni pregresse da soddisfare).

## Descrizione del processo

La settimana corrente scandisce tutte le settimane contenenti ancora dei carichi pendenti e, se sussiste “una condizione necessaria e sufficiente”, prova a soddisfare tali richieste. La priorità maggiore viene assegnata alle lavorazioni più vecchie, per due (o più) lavorazioni “vecchie” che abbiano la stessa “Priority”, vale il principio di ripartizione percentuale cui abbiamo già accennato. Una volta esaurite tutte le lavorazioni pendenti, vengono considerate quelle assegnate alla settimana corrente, sempre considerando le relative “Priority”.

Il termine “condizione necessarie e sufficiente” ha il seguente significato:

* Ogni settimana verifica il suo valore “Late”, e considera soltanto le settimane in coda di attesa che non siano più indietro nel tempo di questo valore.

Se questo requisito è soddisfatto, allora si procede con l’assegnazione in funzione della “Capacity” corrente. Potrà essere assegnata una quantità di lavoro pari alla “Capacity” (nel caso più fortunato), oppure una sua frazione.

* Se la lavorazione non può essere completata nella settimana corrente (abbiamo assegnato tutta la “Capacity”, ma il carico richiesto “Required” non è stato completamente assegnato), allora la settimana pendente rimane in coda di attesa, dopo che il programma ha aggiornato il suo valore “Required” (sottraendone la frazione assegnata).
* Se la lavorazione per la settimana corrente è stata completata, e rimane ancora “Capacity” residua, allora sarà necessario assegnare alla settimana corrente quantità di lavoro richieste nelle settimane successive, e questo in base al valore “Ahead” della settimana. Questo significa che non sarà possibile anticipare la lavorazione di settimane che sono più avanti nel tempo di WEEK\_PLAN + Ahead.

In generale una settimana WEEK è elaborabile se è compresa nel seguente range temporale:

**WEEK\_PLAN - Late <= WEEK <= WEEK\_PLAN + Ahead**

# Implementazione dell’algoritmo di ottimizzazione

Il programma riceve in input una tabella, contenente un certo numero di record, il cui formato è descritto nel paragrafo [Dati di Input e Output](#_Il_punto_di).

Inizialmente, nella tabella di input è necessario operare un raggruppamento per i seguenti campi:

1. PLAN\_BU
2. FLAG\_HR
3. PRODUCTION\_CATEGORY

Questo permette di ottenere l’insieme di record sui quali deve essere applicato l’algoritmo di ottimizzazione.

Poi:

1. Ordinare il dataset ottenuto per WEEK\_PLAN crescente.
2. Ordinare il risultato per “Priority” crescente (il numero più basso significa priorità più elevata).

A questo punto otteniamo una sequenza di record ordinati esattamente nel modo in cui devono essere gestiti.

Avremo perciò i record ordinati nel seguente modo:

PLAN\_BU -> FLAG\_HR -> PRODUCTION\_CATEGORY -> WEEK\_PLAN -> Priority

Il processo di ottimizzazione leggerà in sequenza il set di record costruito in questo modo. Esso si svolge nel modo seguente (si consideri il pseudo codice che segue come una descrizione concettuale e non testuale del codice finale).

## Procedura di ottimizzazione (OptimizeWorkload)

**OptimizeWorkload**

{

// PrepareOptimization sorta la tabella dei

// record in input per:

// PLAN\_BU, FLAG\_HR, PRODUCTION\_CATEGORY,

// WEEK\_PLAN(crescente), Priority(crescente)

// produce in output una tabella denominate

// SortedTable

**PrepareOptimization** -> **SortedTable**

foreach(PLAN\_BU in **SortedTable**)

{

foreach(FLAG\_HR)

{

foreach(PRODUCTION\_CATEGORY)

{

foreach(WEEK\_PLAN)

{

**ElaborateWeek(week\_records)**

}

}

}

}

}

## Procedura che elabora la settimana corrente (ElaborateWeek)

Riceve in argomento la lista dei record relativi alla week, già ordinati per priorità decrescente

**ElaborateWeek(week\_records)**

{

if (**currentweek**.Capacity >0)

{

// prima elabora le richieste in attesa, poi

// passa alla settimana corrente (cioè quella relativa

// ai record week\_records)

if (**GetWaitingRequests**) // waitlist. Esistono settimane in coda di attesa

{

foreach (week in **waitlist**)

{

// elabora le settimane pregresse.

// Fra le settimane in coda di attesa,

// estrae tutte quelle che

// sono compatibili con il parametro “Late”

// della settimana corrente

if (Esistono richieste pregresse)

{

// Elabora le richieste delle settimane

// precedenti. Se è possibile soddisfarle

// solo in parte, aggiorna con la

// frazione non ancora allocata, e vai avanti

**ElabWaitingRequests**

**currentweek**.Capacity -= frazione\_assegnata

}

}

}

// elabora le richieste relative ai record della settimana corrente

**ElabCurrentWeekRequests(week\_records)**

// se rimane capacità residua,

// anticipa la lavorazione delle settimane successive

if (**currentweek**.Capacity > 0)

{

if (**GetAheadRequests**) // aheadlist. Settimane elaborabili in anticipo

{

Foreach (week in **aheadlist**)

{

**ElabAheadRequests**

**currentweek**.Capacity -= frazione\_assegnata

**if (currentweek**.Capacity == 0)

**break**

}

}

}

}

else

{

// solo per le richieste relative alla settimana attuale

// prenota il record per l’elaborazione “Late”

// accoda la richiesta per l’elaborazione successiva

**EnqueueRequests**

}

}

## Procedura Filtra Richieste pendenti (GetWaitingRequests)

Riceve in argomento il parametro WEEK\_PLAN e il parametro Late relativo alla settimana in elaborazione.

In base a queste due informazioni, accede alla lista dei record che rappresentano le settimane il cui carico di lavoro è in attesa di essere servito (oppure è stato solo parzialmente servito), e ricava una lista dei codici WEEK\_PLAN dei record dei quali è permessa l’elaborazione.

Si ottiene così la lista di tutti i WEEK\_PLAN che soddisfano alla regola “Late” ordinata anche in senso temporale, cioè dalla richiesta più anziana, a quella più recente.

Ritorna la lista **waitlist** così ottenuta.

## Procedura di Elaborazione delle Richieste in Attesa (ElabWaitingRequests)

La procedura elabora le richieste pendenti in attesa, e cerca di assegnare i carichi di lavoro alla settimana corrente.

Riceve in input un record che può essere elaborato (un elemento della lista **waitlist** che è già stato validato), e l’oggetto che rappresenta la settimana corrente.

Sottolineo che questa procedura non esegue la validazione delle richieste in attesa (in conformità con la regola “Late”) che è già stata eseguita nella procedura **GetWaitingRequests**.

Sia **waitrecord** il record passato in input del carico di lavoro che può essere assegnato alla settimana corrente, indicata come **currentweek**.[[2]](#footnote-2)

**ElabWaitingRequests(waitrecord)**

{

if (**currentweek**.Capacity > 0)

{

if (**waitrecord**.Required <= **currentweek**.Capacity)

{

**currentweek**.Capacity -= **waitrecord**.Required

**waitrecord**.Required = 0;

aggiorna oggetto in **waitlist**

aggiorna oggetto corrente **currentweek**

}

else

{

**waitrecord**.Required -= **currentweek**.Capacity

**currentweek**.Capacity = 0;

aggiorna oggetto in **waitlist**

aggiorna oggetto corrente **currentweek**

}

}

else

{

end

}

}

## Procedura di Elaborazione delle Richieste Correnti (ElabCurrentWeekRequests)

La procedura è analoga alla precedente, ma lavora soltanto sulla richiesta proveniente dal record corrente. Nel loop di elaborazione di ElaborateWeek vengono scanditi uno ad uno tutti i record relativi alla settimana corrente.

**ElabCurrentWeekRequests(week\_records)**

{

priorities = select Priority,count(\*) from week\_records

Order by Priority asc group by Priority,count

// Le richieste pendenti devono essere soddisfatte raggruppandole per priorità.

// A parità di priorità, le lavorazioni devono essere assegnare mantenendo

// la percentuale reciproca delle richieste

// è requisito che Capacity sia uniforme nella week

initialcap = Capacity;

cap = initialcap;

allocated = 0;

foreach(record in priorities)

{

totreq = Sum(**week\_records** where Priority == record.priority select Required)

foreach(element in **week\_records** where Priority == record.priority)

{

// la ricerca viene eseguita in ordine di priorità dei

// record che richiedono di assegnare un carico di lavoro

// alla settimana corrente, a partire da quello a

// priorità più elevata.

if(cap > 0)

{

toallocate = initialcap \* el.Required / totreq;

if (toallocate <= cap)

{

cap -= toallocate

Required -= toallocate

Allocated += toallocate

}

else

{

Required -= cap;

allocated += cap;

cap = 0;

}

}

}

{Aggiorna i record della week}

}

}

## Procedura Filtra Richieste Anticipabili (GetAheadRequests)

Riceve in argomento la lista completa delle settimane che seguono la settimana corrente, e l’oggetto che rappresenta la settimana corrente.

Dalla lista ricevuta in input seleziona soltanto i record che rappresentano i carichi di lavoro che possono essere elaborati nella settimana corrente (quelli che soddisfano alla regola “Ahead”), ordinati per priorità crescente (la priorità più elevata è rappresentata dal numero più basso).

Ritorna una lista **aheadlist** contenente i record così filtrati.

## Procedura di Elaborazione delle Richieste Anticipabili (ElabAheadRequests)

La procedura elabora le richieste anticipabili, e cerca di assegnare i carichi di lavoro alla settimana corrente.

Riceve in input un record che può essere elaborato (un elemento della lista **aheadlist** che è già stato validato), e l’oggetto che rappresenta la settimana corrente.

Sottolineo che questa procedura non esegue la validazione delle richieste anticipabili (in conformità con la regola “Late”) che è già stata eseguita nella procedura **GetAheadRequests**.

Sia **aheadrecord** il record passato in input del carico di lavoro che può essere assegnato alla settimana corrente, indicata come **currentweek**.

**ElabAheadRequests(aheadrecord)**

{

if (**currentweek**.Capacity > 0)

{

if (**aheadrecord**.Required <= **currentweek**.Capacity)

{

**currentweek**.Capacity -= **aheadrecord**.Required

**aheadrecord**.Required = 0;

aggiorna oggetto in **aheadlist**

aggiorna oggetto corrente **currentweek**

}

else

{

**aheadrecord**.Required -= **currentweek**.Capacity

**currentweek**.Capacity = 0;

aggiorna oggetto in **aheadlist**

aggiorna oggetto corrente **currentweek**

}

}

else

{

end

}

}

## Procedura di Accodamento delle nuove Richieste Pendenti (EnqueueRequests)

Quando una richiesta viene elaborata, l’esito può essere:

1. Richiesta non soddisfatta (Capacity == 0, e quindi non è permessa alcuna assegnazione)
2. Richiesta soddisfatta parzialmente (0 < Capacity < Required)
3. Richiesta soddisfatta totalmente (Capacity >= Required)

Nei casi 1 e 2 è necessario inserire la richiesta fra quelle pendenti, che saranno soddisfatte in una iterazione successiva.

Questa procedura prevede che venga aggiornato il valore “Required” del record e, una volta verificate le condizioni necessarie, esso venga inserito nella coda delle elaborazioni pendenti, per un trattamento successivo.

## Generazione del tracciato di output

L’obiettivo del programma è sostanzialmente quello di compilare nel dataset originale i campi “ALLOCATED” e TCH\_WEEK, che contengono rispettivamente la quantità di ore lavorative allocate per una data settimana, e il codice rappresentante la settimana in cui la data lavorazione sarà completata.

Abbiamo concordato comunque di operare nel seguente modo.

Il dataset originale rimarrà immutato. Ad ogni nuova assegnazione del tempo lavorativo, sarà accodato al dataset originale un nuovo record, contenente l’aggiornamento relativo alla settimana in elaborazione. Per esempio (sono riportati soltanto i campi essenziali):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | TCH\_WEEK | WEEK\_PLAN | Ahead | Late | Priority | Capacity | Required | ALLOCATED | NOT\_ALLOCATED |
| 4 | \_ | 201712 | 8 | 6 | 2 | 0 | 0,209 | 0 | 0,209 |
| 5 | \_ | 201713 | 8 | 6 | 2 | 0 | 1,465 | 0 | 1,465 |
| 6 | \_ | 201714 | 8 | 6 | 2 | 1 | 1,465 | 0 | 1,465 |
| 7 | \_ | 201715 | 8 | 8 | 2 | 1 | 1,465 | 0 | 1,465 |
| 1000 | 201712 | 201714 | 8 | 6 | 2 | 1 | 1,465 | 0,209 | 0 |
| 1001 | 201713 | 201714 | 8 | 6 | 2 | 0,791 | 1,465 | 0,791 | 0,674 |
| 1002 | 201713 | 201715 | 8 | 6 | 2 | 1 | 1,465 | 0,674 | 0 |
| 1003 | …. | 201715 | 8 | 6 | 2 | 0,326 | 1,465 | …. | ….. |

In cui si può osservare che la settimana 201712 viene completamente allocata nella settimana 201714, mentre la richiesta della settimana 201713 viene parzialmente assegnata sempre alla 201714, lasciando una percentuale non allocata di 0,674. La 201713 viene poi completamente allocata nella settimana 201715.

Notare che le righe relative ai record originali non sono state modificate, mentre tutti gli aggiornamenti sono stati aggiunti in append.

Notare che al record 1001 la “Capacity” della settimana 201714 è stata aggiornata, sottraendo dalla “Capacity” iniziale, la quantità allocata alla settimana 201712.

Come risultato della elaborazione sarà possibile restituire anche soltanto il delta, cioè non sarà necessario ritornare anche tutti i record originali, ma soltanto quelli inseriti durante l’elaborazione

## Flusso del processo di ottimizzazione

Di seguito è riportato il flow chart del processo di ottimizzazione descritto nei paragrafi precedenti.



1. Notare che Ahead e Late forniscono il range temporale (dinamico) all'interno del quale si deve svolgere il processo di ottimizzazione [↑](#footnote-ref-1)
2. N.B. Le richieste pendenti devono essere soddisfatte raggruppandole per priorità. A parità di priorità, le lavorazioni devono essere assegnare mantenendo la percentuale reciproca delle richieste. [↑](#footnote-ref-2)