OPTAMS
Optimizer All Monitoring System

Progetto di RICERCA E/O SVILUPPO MURST

ANALISI DEI REQUISITI GENERALI DELL' OTTIMIZZATORE

VERSIONE: 1.2
DATA: 28/12/01

Dipartimento di Sistemi e Informatica
Via Santa Marta, 3
Firenze
Tel: 055 4796365

AUTORI: disit lab, www.disit.org

Part 1. OPTAMS research project

1. Introduction

"Job Shop" problem: a scheduling optimiser

The job shop problem can be regarded as a global model for oganizing the activities of production systems. In the model each *job* is characterised by a technological cycle that needs the intervention of several different resources. The order in which the operations on the resources have to be performed varies from a *job* to another and the resource on which each operation must be realised is previously assigned but the resource can be a class of resources insteat of a uniquely defined. In current industrial reality, characterised by systems with a big number of resources and final products (and consequently a big number of variables involved), the problem of the production "scheduling" or optimisation is really complex (NP-hard). The use of a computerised support is in these cases an essential tool both for the solution of the scheduling problem and for the related optimisation.
Recently, new optimisation techniques have been invented in order to reduce the time to produce suitable solutions. These are typically sub-optimal but are produced with a very short time of computation. These innovations have permitted big improvements in the quality of solutions and in the veloicty of their production. They permitted to manage problems that before could not be considered as the *flexible job shop problem* or the *resource constrained project scheduling*. Some examples of the innovative algorithms recently developed include the *simulated annealing*, the *taboo search*, the *genetic algorithms*, and the *genetic local search*. The strategies that are adopted by these techniques are called *local research techniques*. They allow piloting a short-sightedness algorithm toward the optimisation accepting also behaviours that do not improve the current solution. The first two are substantially short-sightedness heuristic techniques, that is, based on the assessment of solutions in the immediately closeness of the solution at the current iteration; whereas the third uses a particular philosophy derived from mechanisms of biological reproduction and selection that are difficult to be placed in a classical scheme of classification of the heuristic procedures. The last technique mentioned (also defined *memetic search*) is an improvement of the genetic algorithms for what concerns the quality of the solution.

The planner performs the task of optimisation of the phases of the jobs taking into account all the constraints that are present (in the rest of the document the terms "job" and "operation" can be used with the same meaning of phase, subjob, in general). When a plan is ready, it presents the results as a list in which all phases that have to be performed are contained (the so called schedule, schedula). The optimisation is given from the order in which the different phases appear in the list: the first phase will be used as first, the second as second and so on. The optimiser must manage the traffic of files and pieces that is generated by analysing the phases as they present in the list and sorting them coherently with the chosen management policies.
Manufacturing industries integrate in their production pipelines different resources: material, personnel, machines tools, robots, etc. The resource management has to be effective to have market competitive costs. For this, off-line resource planning tools are typically used. Most of them are capable to plan and optimise the production of stable production pipelines (the same set of products for several months). When the production is strongly differentiated these tools are unsuitable.
The production of SHELBOX produced several delays and pauses for personnel and machines. This is mainly due to the fragmentation of production in terms of strong diversification and needs of flexibility. Several different parts are produced in the production environment by the same personnel and machines to satisfy the request of personalisation of the furniture for the caravans, motor-homes, mobile homes, etc. To this end, it is very important to perform optimisation and on-line re-optimisation, and to have the possibility to communicate the work to do at the personnel at the right time and via terminals. This allows changing the production order dynamically according to the re-optimisation and thus the dynamic condition of the production. This is presently left to the sensitiveness of the production responsible but the complexity of production is so huge that several delays, pauses, etc., are inevitably produced.

Integration of on-line optimisers and resource managers

To solve these problems**, an on-line optimiser integrated with the resource manager is needed** such in OPTAMS. OPTAMS will integrate an on-line optimisation techniques of DSI partner with the on-line distributed resource manager of SED partner called AMS (All Monitor System). OPTAMS will be based on barcode pens, microterminals and identification keys for managing peoples and CANBUS for managing machines. OPTAMS is collocated among the MRP, MES areas and the level of control related to the needs of the factories that present differentiate productions with

the necessity of frequent re-optimisations. For our knowledge no other products similar to that developed in the project are present.

The peculiarity of OPTAMS is in the total automation of the MES and control levels, including some aspects MRP that are functional for the management of production in an automatic manner. Typically, the MES solutions collect data coming from the field devices and verify their congruence with the adopted planning, dynamically modifying, if necessary, the production plan. From the literature and the market analysis, it has been observed that many other systems cannot order the execution of the different working phases because this task is generally manual or made through paper: the operator has a printing of the production programme generated by the optimiser. Moreover the planning products are not sufficiently flexible to be capable of considering all the constrains used for the direct control of the different resources: precedence, type of machine, groups of operators, competence of personnel, etc.

The main innovations of OPTAMS will consist in the integration of the AMS resource manager with the on-line optimiser build at the state-of-the-art by DSI. The on-line optimiser will be based on Genetic Algorithms or Taboo Search. These are typically the most powerful and satisfactory optimisation techniques that are producing the best results for the considered cases. The selection among these two approaches will be performed during the requirement analysis in which details about the production process of SHELBOX will be clarified. The system will be also endowed with a multilingual support.
In order to obtain an effective reduction of production costs from the information flow about measurements of times, it is necessary to *use* this information to plan the future production to organise appropriately the sequence of the different operations. For example, the transferring of files, the movement of raw pieces, the beginning of the workings, the activation of personnel, etc. This task is performed by an algorithm of planning that optimises such sequence taking into account the different needs (delivery times, availability of resources, errors, variance of workload among resources, general reduction of delivering time, reduction of production costs, exploitation of resources, etc.). If the system is production process presents a considerable number of manufactures and/or the number of different products passing on the same production line is big, then the necessity of an automatic on-line optimisation planner of the production is indispensable to produce the product at competitive costs and prices.
The phases of resource management and optimisation planning are connected but at the same time independent. This means that a good resource management of the production is impossible without a previous planning, nevertheless the management policies of the production pipeline are independent from the adopted planning model.
The optimisation engine will be also capable of supporting the re-optimisation when critical cases occur corrupting the planned schedule. For instance when the efficiency of the actual production decreases due to

- the lack of resources: human or material,
- the accumulation of delay,
- the presence of too large variance of workload among resources,
- the manual request of machine to perform a correction,
- the breaking of a machine or tools, etc.

Also the manual re-optimisation will be possible. The action of optimisation/re-optimisation can be also useful for assessing the impact of changes in the production process. These can be provoked by new products to be produced, new resources, changes in the current production (time, number of pieces, etc.), problems to people and machines, etc.
The control of the work of the personnel will be performed by using the distributed system at the basis of AMS supporting the OPTAMS. This is endowed with industrial computers and microterminals that allows the personnel to read the actions to be performed and to recording their work. This is possible by using bar code pens and special keys that allows at the system to recognise the single person as belonging to a given category of working people in the factory. This will allow to control and trace the production without tracing the single person according to the legal rules.

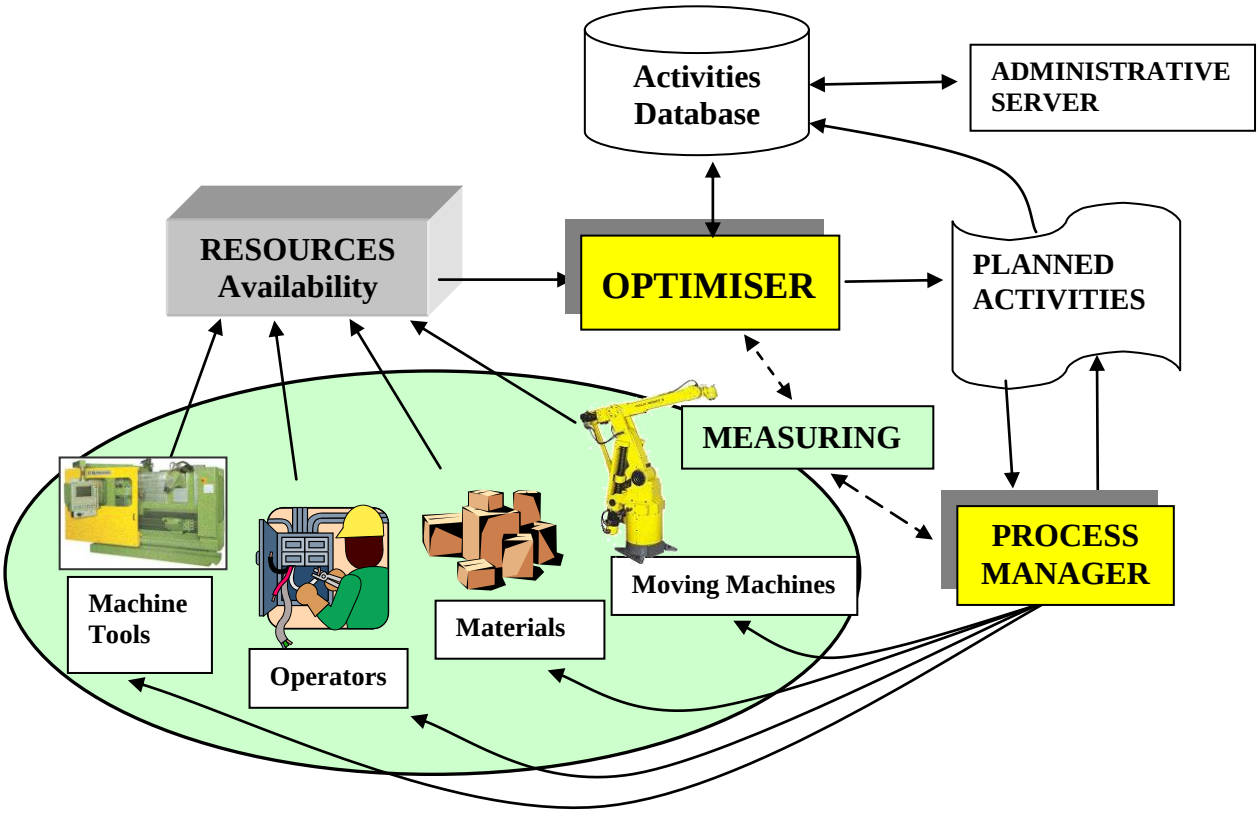
2. Description

The following figure depicts the most important aspects of the optimisation and management processes with related components of the OPTAMS solution. Most of the mentioned components present a direct correspondence with the WPs successively described and describing the whole development plan of the OPTAMS project. In the next sections, details about the general architecture of OPTAMS is reported in terms of computer based machines and software architecture and components.

The resources of the plant are: operators, machines, materials, robot to move the materials and so on. A flexible production activity needs to make a plan to schedule for every resource the jobs to perform and the time assigned. The resources of the plant are configurated in the administrative server, which also receive the information about the future commitments of the plant. All the information about the past and the future commitments reside on the activities database, which can be accessed by the administrative server and optimiser. The Process Manager acts on the production process on the basis of the schedule produced by the Optimiser (Planned Activities) of the jobs assigned to

every resource and an estimation of the time the resource will be occupied in performing such jobs. The schedule of the planned activities for the jobs and the resource assigned to each job can be manually optimised or automatically optimised through the Optimiser to produce a new schedule of the Planned Activities, more efficient in resource management on the basis of specific criteria: minimising delivering time, distributing workload, minimising costs, etc.

The Process Manager according to the Planned Activities, through a data acquisition tool (Measuring) tracks the production while the jobs are in progress, instructing each resource on the next job to perform when the resource finish its previous task. A re-optimisation can be invoked, for example if delays have occurred in the real advancement of the jobs. Then a new schedule (Planned Activity) is produced by the Optimiser taking in account the current situation of advancement of the jobs also in terms of Reasource Availability.
The process manager instructs the resources on the jobs that are planned for them: the instructions on the job to perform can be sent to the machine tools in a completely automated way on the factory LAN, or to the operators through the MICROTERM terminals when the operator check in with the programmable key.



Process components:

- **RESOURCES Availability. T**he resources of the factory are machine tools, personnel, material and moving machines. In some cases some of these resource types can be missing or one type can be dominant with respect to the others. Typically in mainly manual production processes human resources and materials are dominant with respect to machines. The Resource Availability module maintains under control the status of the resources in terms of availability, efficiency, detailed status.

- **OPTIMISER:** the on-line optimiser/re-optimiser that takes into account the Available Resources and their status and measuring. A resource available at a given time instant can be not available during the production process for faults of simple since it has been finished (for material). In some cases, of personnel problem may occur that constrain the person to leave the production process for minutes, hours or days. Therefore, on-line re-optimisation is mandatory. The optimiser produces a list of PLANNED ACTIVITIES on the basis of the Activities Database containing the committed works acquired from the Administrative Server and commercial part of the factory. This activity is directly obtained from AMS product of SED. The committed works are transformed in real activities by the Optimiser on the basis of several criteria: deadlines, available resources, strategic actions of the company, available reserves, etc.

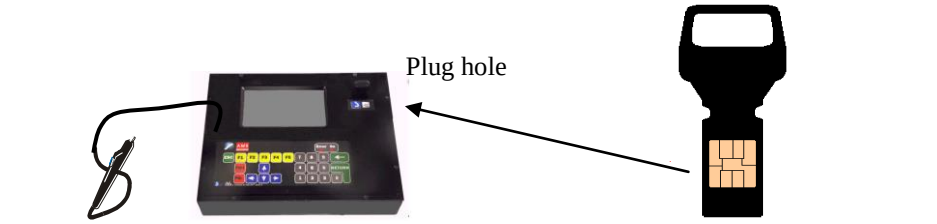
- **PLANNED ACTIVITIES:** the schedule that collect the details of the activities that have been planned to be performed in the next hours, days, weeks, etc. The PLANNED ACTIVITIES are actualised on the basis of the effects of production process at each re-optimisation. The actualisation is performed by the Process Manager and the update is directly performed in the ACTIVITY DATABASE. The updated values are used by the ADMINISTRATIVE SERVER for estimating the real production costs for each product involved in the process.

- **PROCESS MANAGER.** The Process Manager attempts to manage the production according to the PLANNED ACTIVITIES. These can be perfectly followed or not. In real cases, some activities can start with some delay, may finish with some delay. In general, the detailed process activities/task may have a duration different to what has been planned. These changes are recorded according to the MEASURING activity of the process management. When critical changes and variations are recorded then a re-optimisation phase is started.

- **ACTIVITIES DATABASE:** The Activities Database contains the committed works acquired from the Administrative Server and commercial part of the factory. This activity is directly obtained from AMS product of SED. The committed works are transformed in real activities on the basis of several criteria: deadlines, available resources, strategic actions of the company, available storage, etc.

- **ADMINISTRATIVE SERVER.** It collects the administrative aspects and reports the real costs of the production on the basis of measures performed on the production and thus on the basis of the actualised PLANNED ACTIVITIES. It includes and collects the database of the committed works for the production and the factory historical data. The trace of the actually performed activities are stored into this server.

- **MEASURING.** The process of tracking, monitoring the resources and activities of the production process. This is the core of the system. The measuring is performed on people by means of microterminals and industrial computers endowed of laser code pens (for activities and material identification) and electronics keys associated with categories/groups of personnel. The personnel read the description of the work that has to be performed on the microterminal. The same terminal is used to collect comments about the performed activity together with the time for its execution. The measuring on machines is directly performed by using the connection to the machines via CANBUS or PLC. The measures are mainly related to the starting time, ending time, and thus on activity duration.



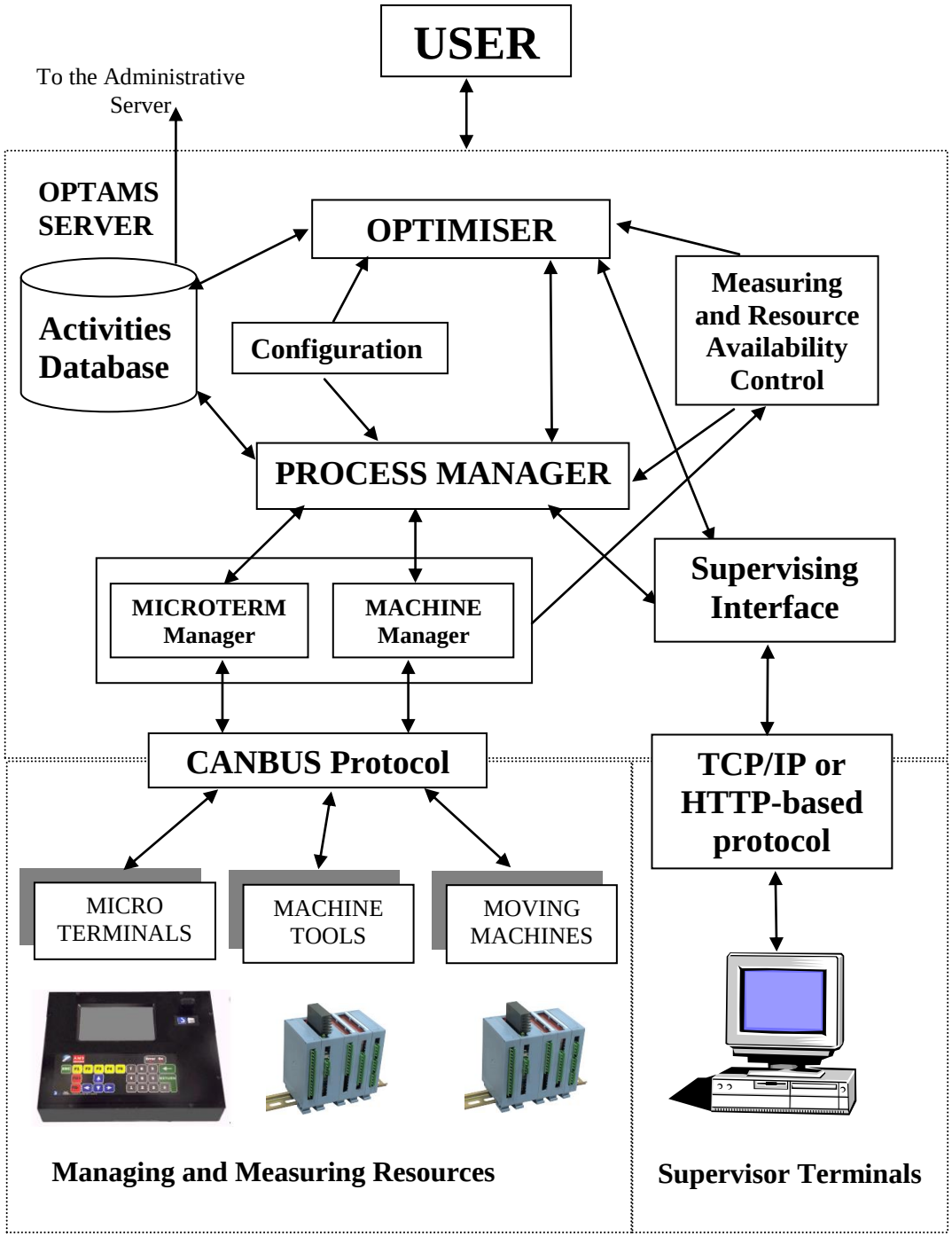
OPTAMS Software Architecture

In the next page, the figure depicting the software architecture of OPTAMS is reported. In the bottom of the figure, the software components to manage terminals/ports that interface the resources of the plant with the OPTAMS server are present: the MICROTERM terminals, the machine tools interfaces and the moving machine interfaces. These are connected via CANBUS protocol with the OPTAMS server, where the Process Manager and Optimiser (for optimisation and on-line re-optimisation) reside. They interact with the Activities Database which holds information on the works committed and all the past and future activities. They also take information from the Measuring Resource Availability Control on the basis of the Configuration imposed during installation.
The Optimiser receives a schedule of the planned activities and optimises it for a better resource management on the basis of the Configuration and Resource Availability. The communications between the operator terminals and the Process Manager are performed via CANBUS where the resources communicate to the server their own status, the termination of the job performed, and receive instructions on the next job to perform. The process manager can track the production and performs a time check based on the information exchanged with the terminals.
For these reasons the software architecture is mainly divided in OPTAMS Server, Management and Measuring Resources, and Supervisor Terminals.

- **OPTIMISER:** the on-line optimiser/re-optimiser takes into account the Available Resources and their status. A resource available at a given time instant can become not available during the production process for faults of simple since it has been finished (for material). Personnel problems may occur to constrain operator to leave the production process for minutes, hours or days. In some cases, the on-line re-optimisation is mandatory to optimise

the production. This has to be also performed when resources are finished or when a machine is manually put off-line. The optimiser produces a list of PLANNED ACTIVITIES on the basis of the activities database containing the committed works acquired from the Administrative Server of the factory. The committed works are transformed in real activities by the optimiser on the basis of several criteria: deadlines, available resources, strategic actions of the company, available reserves, etc.

- **Configuration:** is the static information available on the resource physical location in the plant: their allocation, position, efficiency, productivity, cost, etc.
- **MICROTERM and MACHINE Managers:** the processes to send messages to resource to command their activity and to track, monitoring the resources and activities of the production process. These processes are used by the Process Server to send messages to the terminals and machines and by the Measuring and Resource Availability Control module for the assessing the resource status and detailed aspects.
- **CANBUS Protocol:** an high level protocol based on the CANBUS standard. It will allow the communication of both low-level (I/O control signals) and high-level bidirectional messages and information coming from more sophisticated terminals and microterminals.
- **Measuring and Resource Availability Control.** The measuring is performed on people by means of microterminals and industrial computers endowed of laser code pens (for activities and material identification) and electronics keys associated with categories/groups of personnel. The machine measuring is directly performed by using the connection to the machines via CANBUS. The measures are mainly related to the starting time, ending time, and thus on activity duration. The resources of the factory are machines tools, personnel, material and moving machines. In some cases some of these resource types can be missing, in other factory one type can be dominant with respect to other. Typically in mainly manual production processes human resources and materials are dominant with respect to machines. During the resource measuring also the resource status is performed in order to track the resource availability and reporting eventual alarms on the OPTAMS Server.
- **PROCESS MANAGER:** The Production Manager attempts to manage the production according to the PLANNED ACTIVITIES defined by the Optimiser. These can be perfectly followed or not. In real cases, some activities can start with some delay, may finish with some delay and may have duration different to what has been planned. These changes are recorded by the Measuring and Resource Availability Control process. The Process Manager takes into account of the actual values measured on the production process such as changes in starting and ending time, exploitation of resources, etc. When critical changes and variations are recorded then a re-optimisation phase is activated needed. Planned and real measures are used by the Optimiser to perform or not a re-optimisation and for communicating actual results to the production manager.
- **Activities Database:** The Activities Database contains the committed works collected in the Administrative Server and commercial part of the factory. This activity is directly obtained from AMS product of SED. The committed works are transformed in real activities by the Optimiser on the basis of several criteria: deadlines, available resources, strategic actions of the company, available reserves, etc.
- **Supervising Interface**. It allows to export information and/or the user interface of the Optimiser and of the Process Manager on the Supervisor Terminals. They are distributed on the factory in order to monitor the control process from a distributed network of stations and not only from the OPTAMS Server. These Supervising Terminals can be also used to browse the work that has to be done in the next future.
- **TCP/IP or HTTP-Based Protocol**. To allow the Supervising a suitable high level protocol will be performed. To remote the interface of the Optimiser and Process Manager Intranet technology as well as more specific tools (such as PC Any ware or RADMIN) will be considered. This approach will be useful to take the control of the plant even via a remote connection. For example, via modem. The Production Line associated with a Supervisor Terminal can be endowed of one more WEB Cameras for monitoring in live the activity.

- 

The following part illustrate the optimization model of the OPTIMISER module.

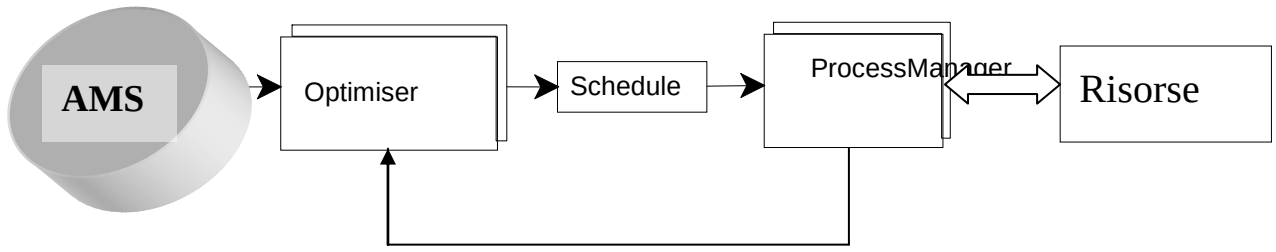
Part 2. OPTIMISER : il modello di ottimizzazione

1. Introduzione (importante)

Questo documento riporta la descrizione del processo produttivo e del modello di ottimizzazione per la pianificazione di processo sulla base del sistema di gestione delle commesse AMS.
Per politiche di gestione si intende il modo con il quale viene svolto il transito all'interno dell'azienda degli articoli che devono essere lavorati e di tutte le informazioni ad essi pertinenti come l'evoluzione dello stato dei semilavorati e delle risorse nel ciclo di produzione. L'integrazione totale delle risorse presenti è difficilmente ottenibile in assenza di un'adeguata circolazione e regolamentazione delle informazioni inerenti le stesse. La politica di gestione deve, dunque, farsi carico di tali prerogative.
Per rendere possibile la gestione di tale flusso informativo è necessario organizzare opportunamente la sequenza delle varie operazioni da eseguire quali, ad esempio, il trasferimento di informazioni a macchine e a persone etc.; questo compito deve essere svolto da un algoritmo di pianificazione che ottimizzi tale sequenza tenendo conto delle varie esigenze (tempi di consegna, disponibilità delle risorse, errori di varia natura). Qualora il sistema sia composto da un numero esiguo di macchine e/o gli articoli da realizzare siano in numero limitato è possibile fare a meno di una pianificazione di tipo automatico. Nel caso in cui venissero a mancare le ipotesi precedenti (il numero di risorse non è trascurabile o la quantità degli oggetti da realizzare e le loro fasi di lavorazione sono considerevoli) si impone la necessità di procedere ad una pianificazione automatica della produzione con relativa ottimizzazione della gestione delle risorse.
Le fasi di gestione e pianificazione sono intrinsecamente legate tra di loro, ma allo stesso tempo risultano indipendenti. Con ciò si vuole sottolineare il fatto che non può esservi una buona gestione delle risorse coinvolte nella produzione senza una sua preventiva programmazione/pianificazione, e la politica di gestione del ciclo di produzione dell'azienda risulta del tutto indipendente dal modello di pianificazione adottato. Il seguente disegno riassume i concetti precedentemente esposti.

Il modulo ottimizzatore, denominato *Optimiser* riceve in ingresso informazioni di vario tipo sugli ordini inseriti nel database AMS. Il database AMS contiene informazioni che consistono essenzialmente in:

- deadline entro la quale l'ordine deve essere eseguito
- creazione delle commesse dagli ordini
- esplosione delle commesse in semilavorati
- descrizione degli articoli che l'azienda è in grado di produrre
- fasi di produzione caratteristiche dell'azienda
- descrizione delle risorse aziendali



Il pianificatore svolge il compito di ottimizzare le fasi dei sottolavori tenendo conto di tutti i vincoli che sono presenti fra le fasi di lavorazione e le risorse; quando ha terminato presenta i suoi risultati in una lista (schedula) nella quale sono contenute tutte le fasi che debbono essere eseguite, le risorse che devono eseguirle e il momento nel quale tali fasi devono essere eseguite; l'ottimizzazione è data dall'ordine nel quale compaiono le varie fasi nella lista: la prima fase dovrà essere eseguita per prima, la seconda per seconda e così via. L'ottimizzazione cerca di minimizzare alcuni fattori di costo al fine di produrre una pianificazione "ottima" in base ai fattori definiti.

2. Stato dell'arte nei modelli di ottimizzazione

2.1 Introduzione

Nella attuale realtà industriale, caratterizzata da impianti con un elevato numero di macchinari e prodotti realizzati (e, conseguentemente, da numerose variabili in gioco) il problema della "schedulazione" della produzione, di natura sostanzialmente combinatoriale, risulta essere molto complesso (NP-hard). L'utilizzazione di un supporto informatico si pone in questi casi come uno strumento essenziale sia per la soluzione del problema di scheduling, che per la relativa ottimizzazione. La necessità di ottenere buone soluzioni in tempi brevi ha spinto molti ricercatori a studiare nuove tecniche di ottimizzazione combinatoriale di tipo euristico; tale approccio è probabilmente oggi il più adatto per affrontare i problemi di programmazione operativa più complessi, in quanto consente di adottare modelli sufficientemente realistici e privi di eccessive semplificazioni, mantenendo contemporaneamente entro limiti ragionevoli i tempi di calcolo necessari a risolvere il problema. In figura [5] sono mostrate le tecniche che sono state applicate in letteratura per risolvere problemi di schedulazione di natura combinatoria (si fa riferimento in particolare al problema noto come *job shop* o Π_j, definito formalmente nel prosieguo).

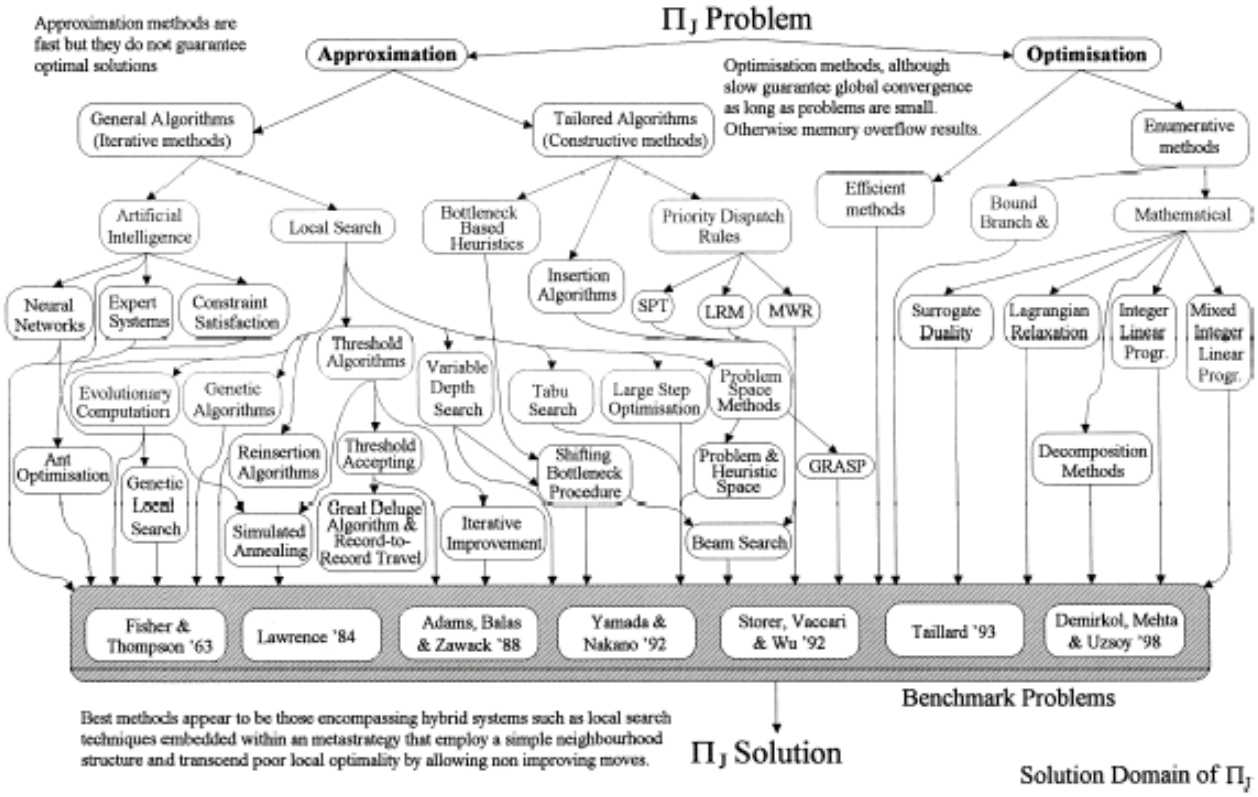


Fig. 1. The phases of Π_x research.

Storicamente le prime tecniche che sono state applicate per risolvere questi problemi sono state tecniche di tipo euristico; i pionieri in questo campo sono stati ricercatori quali Johnson (1954) , Ackers (1956) e Jackson (1956). L'applicazione di semplici regole permetteva di risolvere problemi di modesta complessità. Negli anni 60' l'attenzione si spostò sulla ricerca di soluzioni mediante algoritmi che formulano i problemi in termini di programmazione matematica. Un esempio di approccio risolutivo di questo tipo è presentato in [7] (una descrizione esauriente delle tecniche di programmazione lineare può essere trovata in [33]). Nonostante il notevole livello teorico delle ricerche tale strategia non ha mai prodotto risultati entusiasmanti soprattutto perche' non si sono rivelati capaci di trattare problemi aventi dimensioni rilevanti. Il metodo più utilizzato tra questi ultimi è il *Branch and Bound* che adotta una strategia di tipo enumerativo costruendo dinamicamente un albero in cui sono implicitamente presenti le soluzioni. Questa metodologia di ricerca produce ottimi risultati in presenza di problemi di piccolo taglio, ma si rivela praticamente inutilizzabile non appena la dimensionalità del problema aumenta a causa dell'eccessivo carico computazionale che

viene richiesto. Vi è da notare che recentemente sono state proposte alcune varianti di questa tecnica che tendono ad eliminare preliminarmente la maggior parte dei nodi dell'albero che portano a soluzioni errate [1], [3] e [8]. Durante i successivi anni 70' e nella prima metà degli anni 80' sono state riprese in considerazione strategie di ricerca basate su metodi di approssimazione in luogo delle metodologie volte alla programmazione matematica esatta. Sebbene le metodiche che usano metodi approssimati non siano, il più delle volte, in grado di trovare la soluzione ottima esse mostrano un incontestabile vantaggio per quanto riguarda la velocità di ricerca rendendo così trattabili problemi più complessi da risolvere. I primi algoritmi di approssimazione che nacquero furono quelli basati su regole a priorità di assegnamento (*pdrs, priority dispatch rules*). Durante tale periodo furono create un una gran numero di regole differenti; il vantaggio maggiore insito nell'utilizzare tale approccio è dato dal fatto che tali algoritmi sono piuttosto semplici da realizzare e presentano un esiguo costo computazionale. D'altro canto, a causa della loro miopia nella ricerca, dovuta al fatto che considerano le soluzioni solamente allo stato corrente o al massimo nelle immediate vicinanze, la qualità delle soluzioni degrada rapidamente all'aumentare delle dimensioni del problema.
Tra il 1986 ed il 1991 sono state inventate nuove tecniche o ne sono state sviluppate alcune i cui prodromi già esistevano negli anni precedenti. Tali innovazioni hanno consentito di fare notevoli passi avanti in termini di qualità delle soluzioni e di velocità della ricerca. Soprattutto hanno permesso di trattare problemi assai complessi che in precedenza non permettevano di essere considerati quali il *flexible job shop problem* o il *resource constrained project scheduling*. Alcuni esempi degli algoritmi innovativi sviluppati recentemente includono la *simulated annealing* (SA), la *Tabu search* (TS), gli *algoritmi genetici* (AG), e la *genetic local search* (GLS); Le strategie che vengono adottate da queste tecniche sono dette di *ricerca locale*. Esse permettono di pilotare un'algoritmo miope verso l'ottimalità accettando anche mosse che non migliorano la soluzione corrente.
Le prime due sono sostanzialmente tecniche euristiche miopi, basate cioè sulla valutazione delle soluzioni nelle immediate vicinanze della soluzione alla iterazione corrente, mentre la terza utilizza una filosofia molto particolare derivata da meccanismi della riproduzione e selezione biologica, difficilmente collocabile in un classico schema di classificazione delle procedure euristiche. L'ultima tecnica menzionata (definita anche *memetic search*) non è altro che un miglioramento degli algoritmi genetici soprattutto in termini di qualità della soluzione. In linea di massima nei lavori presenti in letteratura le tecniche ivi descritte sono realizzate coniugandole opportunamente con euristiche ad hoc per il problema di interesse. A tal proposito vi è da notare come, in realtà, sia assai difficile classificare sistematicamente le tipologie di problemi che possono presentarsi nell'ambito dei sistemi produttivi reali. Altrettanto complesso risulta, perciò, elencare le metodiche risolutive applicate nell'uno o nell'altro caso. Al fine di operare un confronto tra le tecniche che al momento appaiono più promettenti (TS, AG e SA) è utile descrivere come viene affrontato uno dei più classici problemi della ricerca operativa quale il *job shop problem*. Partendo dalla risoluzione di questo problema è possibile poi estenderne i risultati per trattare pianificazioni che richiedono problematiche di maggior complessità quali quelle che rientrano nella generica categoria FMS [7], [8], [24] (*flexible manufacturing systems*). Un'ultima precisazione è inerente la natura dei parametri produttivi considerati per affrontare i problemi di schedulazione connessi. Tali valori (tempo stimato di processamento, deadlines, numero di risorse disponibili, etc.) sono stati trattati come deterministici. Alcuni ricercatori (Luh, Chen e Thakur) [6] hanno proposto algoritmi in grado di operare in presenza di parametri di produzione aleatori; sebbene i risultati siano incoraggianti se applicati a problemi di piccolo taglio si ha l'impressione che si debba procedere ancora ad una maggiore sistematizzazione formale del problema.
L'obiettivo prefissato, dunque, è quello di effettuare un confronto tra le tecniche, applicate a problemi di programmazione operativa della produzione di sistemi produttivi che operano prevalentemente su commessa, alfine di definire un modello per il problema specifico.

2.2 Il problema del job shop

Un modello di sistema produttivo fra i più completi è quello noto in letteratura con il nome di *job shop*. In questo sistema ogni *job* è caratterizzato da un ciclo tecnologico che richiede l'intervento di più macchine/risorse diverse; l'ordine con cui vanno eseguite le operazioni sulle macchine è diverso da *job* a *job*, e la macchina su cui ciascuna operazione deve essere realizzata è assegnata a priori.
Il problema può essere formalizzato in maniera più rigorosa nel modo seguente.
Sono dati:

- Un insieme di lavori J_i, (i=1,…,N)
- Un insieme di macchine m_j, (j=1,…,M)
- Per ciascun lavoro J_i è data una sequenza di operazioni che costituiscono il suo ciclo di lavorazione: con o_{ik} si indica la k-esima operazione (k=1,…,K_j) del lavoro J_i.
- Per ciascuna operazione o_{ik} è assegnata la macchina capace di eseguire tale lavorazione ed il relativo tempo di processo p_{ik}.

Nei prossimi paragrafi saranno descritte le applicazioni delle tre tecniche euristiche precendentemente citate a tale problema, assumendo come obiettivo della schedulazione la minimizzazione del *makespan* (tempo di completamento dell'ultima operazione schedulata).

2.3 Tabu Search (TS)

Il Tabu-search è stato ideato da Fred Glover nel 1986 [15], [30]. Tale algoritmo iterativo esplora un insieme di soluzioni possibili del problema facendo mosse ripetute da una soluzione x ad un'altra x_n appartenente all'intorno N(x) di x.
L'intorno N(x) della soluzione corrente è individuato dall'insieme delle soluzioni, dette soluzioni candidate, che possono essere raggiunte, a partire da x, eseguendo una "mossa" (ad esempio: scambio di due o più operazioni assegnate alla stessa macchina).
La caratteristica principale del TS è l'introduzione di memoria nel processo di ricerca della migliore soluzione. La struttura di memoria consiste in una lista di mosse, chiamata *lista Tabu*, che non possono essere eseguite all'iterazione corrente per passare da una soluzione alla successiva. Le mosse appartenenti alla lista Tabu sono mosse che sono state eseguite recentemente (memoria *recency)* o che sono state effettuate frequentemente nelle ultime iterazioni (memoria *frequency*).
La struttura di memoria impedisce quindi all'algoritmo di rimanere intrappolato in minimi locali guidandolo verso lo spazio delle soluzioni ancora inesplorate, permettendo di selezionare anche mosse che portano a soluzioni caratterizzate da un valore della funzione obiettivo maggiore di quello della soluzione corrente.
Lo stato Tabu di una mossa può essere violato se questa soddisfa il *criterio di aspirazione*, ovvero se il valore della funzione obiettivo ad essa associato è il minore di tutti quelli determinati sino alla soluzione corrente (criterio di aspirazione standard).
La procedura si interrompe quando viene raggiunto un numero di iterazioni massimo fissato a priori.

Applicazione della TS al problema del job shop

Una delle implementazioni più efficaci della tecnica di *Tabu search* per il problema del job-shop è quella considerata in [25]. Nel quale viene proposto un algoritmo TS per il problema del JS che produce ottime prestazioni sia dal punto di vista dei tempi di calcolo che della qualità della soluzione.

Intorno della soluzione corrente e modalità di esecuzione della mossa

L'intorno della soluzione corrente può essere determinato con le stesse modalità utilizzate nel *simulated annealing*; In tal modo può essere possibile effettuare un confronto tra le due tecniche in termini di capacità di guida del processo di ricerca della migliore soluzione. Oppure possono essere considerati altri metodi di ricerca dell'intorno della soluzione corrente come già esplicitato a proposito del *simulated annealing*.
Per quanto riguarda la modalità di esecuzione della mossa, ovvero il passaggio da una soluzione alla successiva, questo, nella descrizione delle caratteristiche di base della tecnica, differisce completamente dal SA.
Esso consiste, infatti, nel selezionare, come nuova soluzione, quella appartenente all'intorno della soluzione corrente, caratterizzata da un valore della funzione obiettivo minore delle altre soluzioni candidate e non appartenente alla *lista Tabu*. La restrizione Tabu può essere violata nel caso in cui venga soddisfatto il criterio di aspirazione.
I due modelli di ottimizzazione, SA e tabu, possono anche essere combinati come sara' mostrato in seguito.

Struttura di memoria (creazione e gestione della lista Tabu)

La struttura di memoria, ovvero la *Tabu list*, consiste in una matrice quadrata $TM_{n,n}$, dove n rappresenta il numero complessivo delle operazioni del problema.
Possono essere implementate due tipologie di struttura di memoria diverse indicate con il nome, rispettivamente, di *recency* e *frequency memory*.
Nella memoria *recency*, il generico elemento TM_{ij} rappresenta il contatore dell'iterazione in cui l'arco (i,j) è stato invertito l'ultima volta; l'inversione dell'arco (i,j) è impedita fino a che:

TM_{ij} + *Tenore* > Contatore iterazioni

Dove il *Tenore* rappresenta la lunghezza della lista Tabu, ovvero il numero di iterazioni per cui una coppia di operazioni non può essere invertita. Questo tipo di memoria impedisce di rivisitare soluzioni individuate da mosse recentemente effettuate sulla base del principio di base che caratterizza la tecnica Tabu search:

"*Una mossa, considerata buona (in termini di valore della funzione obiettivo) all'iterazione corrente, lo sarà anche per le successive iterazioni"*.

Nella struttura di memoria *frequency*, invece il generico elemento TM_{ij} rappresenta il contatore del numero di inversioni dell'arco (i,j) nel processo di ricerca: l'inversione dell'arco è impedita quando $TM_{ij}>f$, dove f è un parametro funzione della geometria del problema ed è fissato a priori.

Questa restrizione interviene ogni *Nfrequency* iterazioni avvenute senza miglioramenti ed è quindi utilizzata più come strumento di blocco dei cicli che come una vera e propria procedura di guida dell'algoritmo.
Assume allora importanza fondamentale il valore attribuito al Tenore, ovvero alla lunghezza della lista Tabu: accade infatti che, se questa è troppo piccola, l'algoritmo può rimanere intrappolato in cicli, ovvero rivisita periodicamente le stesse soluzioni; viceversa, se la lunghezza della Tabu list è troppo grande la procedura Tabu è troppo restrittiva. Per individuare allora il corretto valore del Tenore al variare del problema e, nello stesso problema, al variare della fase di ricerca, è possibile utilizzare una lista Tabu dinamica consistente nel variare il valore del Tenore in funzione dei risultati della ricerca. Si ha infatti:

- se il valore della funzione obiettivo è minore del migliore valore incontrato sino adesso, la lunghezza della lista viene posta uguale ad uno;
- se la ricerca è in una fase in cui la funzione obiettivo migliora (rispetto alla soluzione precedente) e la lunghezza della lista è maggiore di una quantità minima, il Tenore viene diminuito di una unità;
- se la ricerca è in una fase in cui la funzione obiettivo peggiora e la lunghezza della lista è minore di una quantità massima, il Tenore viene aumentato di una unità.

Strategie di ricerca e parametri utilizzati

Con questo termine sono stati indicati tutti quegli accorgimenti e procedure che, pur non rientrando nei componenti fondamentali della *Tabu search*, sono necessari per il buon funzionamento dell'algoritmo e per migliorarne le prestazioni.
Le strategie di maggior interesse si basano tutte sull'introduzione di casualità nella procedura di guida dell'algoritmo; si ha infatti:

- quando tutte le mosse sono classificate Tabu e nessuna di esse soddisfa il criterio di aspirazione viene selezionata casualmente una soluzione fra quelle candidate;
- i valori limite, *min* e *max*, della lunghezza della lista Tabu sono estratti casualmente, ogni Λ iterazioni, da due intervalli, rispettivamente (a,b) e (A,B);
- per evitare l'intrappolamento dell'algoritmo in cicli si può inserire una procedura di casualizzazione del tenore;
- la procedura di interruzione del processo di ricerca è basata sul numero massimo di iterazioni (*Maxiter*) effettuate dall'algoritmo senza miglioramenti.
- Tutti i parametri descritti precedentemente (*tenore, min, max, f, Nfrequency* etc…) debbono essere espressi in funzione delle sole dimensioni del problema (N° macchine, N° lavori) allo scopo di conferire un'elevata flessibilità all'algoritmo.

3. Analisi generale (importante)

3.1. Il Problema

Il problema oggetto dell'ottimizzazione è il seguente: l'azienda riceve le varie commesse o lavori dai clienti (denominati anche ordini); ogni commessa può essere costituita da uno o più job (articoli o semilavorati). Tipicamente un ordine deve essere eseguito entro una certa data (deadline). Questo parametro, naturalmente, è da considerarsi di fondamentale importanza nell'economia del problema. Su un articolo vengono definite una serie di operazioni (o task) che devono essere obbligatoriamente eseguite in sequenza. Ad ognuna di queste fasi corrispondono una o più risorse che possono realizzare la suddetta operazione: ad esempio, una fase potrebbe essere eseguita da una certa specifica risorsa appartenente a un certo sottoinsieme delle risorse a disposizione dell'azienda.
L'obiettivo è quello di ottimizzare la pianificazione della produzione degli articoli nel minor tempo possibile, col minor costo, cercando di rispettare i tempi di consegna degli stessi, distribuendo uniformemente il carico di lavoro sulle risorse messe a disposizione dall'azienda. È richiesta notevole flessibilità in quanto malfunzionamenti o procedure errate sono piuttosto frequenti in impianti complessi, pertanto deve essere possibile procedere, in qualsiasi momento, ad un'immediata ripianificazione che tenga conto della situazione contingente.

3.2. Il modello di processo di produzione in OPTAMS

Ogni articolo o semilavorato viene considerato come un job. Una commessa è quindi composta da vari job. Un job può essere suddiviso a sua volta in operazioni elementari denominate tasks, o fasi di produzione. Tali tasks si identificano essenzialmente con le fasi di lavorazione caratteristiche dell'azienda.

3.2.1 Descrizione del "Job"

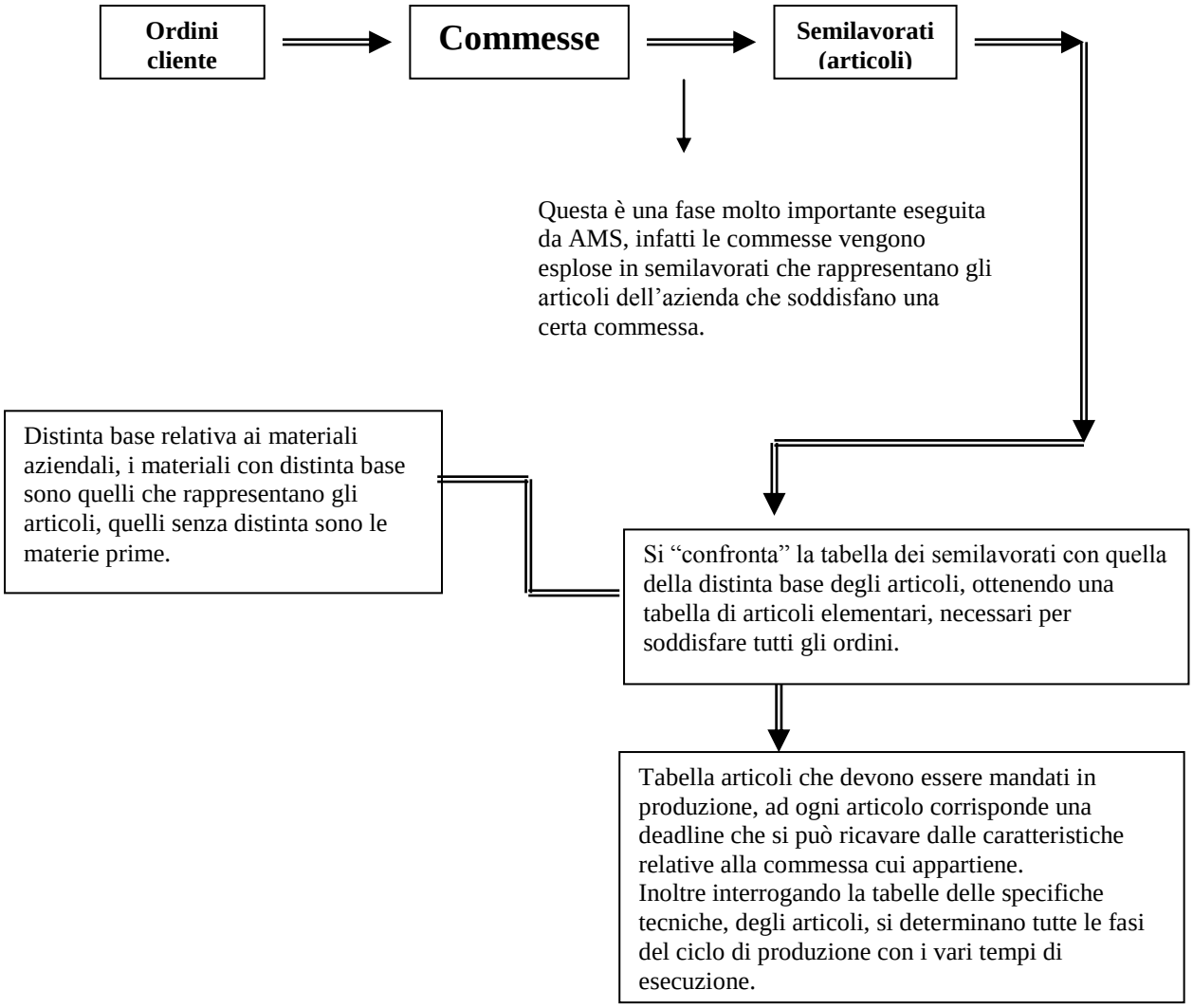
Analizziamo in modo dettagliato a cosa corrisponde un job. In AMS sono inseriti tutti gli ordini con evidenziata la loro deadline: il programma provvede a creare da questi le "commesse" che a loro volta sono esplose in semilavorati.
Ogni semilavorato corrisponde ad articoli che devono essere prodotti dall'azienda o comprati esternamente, ai fini dell'ottimizzazione consideriamo soltanto quelli che devono essere mandati in produzione. Possiamo affermare che un job corrisponde al processo di lavorazione necessario per creare questi articoli.
Il job o processo di produzione è caratterizzato da un insieme di operazioni elementari o tasks che rappresentano fasi di produzione da eseguire nel giusto ordine per portare a termine il lavoro in esame.

3.2.2 Tasks (o fasi di produzione) e relativi vincoli

Le informazioni relative alle fasi di produzione o tasks sono caratteristiche dell'azienda e sono deducibili dal database AMS. Esse sono caratterizzate da un codice_fase e da una breve descrizione di quest'ultima.
Ad ogni articolo corrisponde una scheda, chiamata distinta base, dalla quale si estraggono tutte le informazioni necessarie per portare a termine il processo di produzione dell'articolo stesso.
Queste informazioni sono di due tipi:

- elenco delle fasi da eseguire,
- materie prime necessarie.

Le fasi da eseguire sono poste sotto forma di elenco e in corrispondenza di ogni fase abbiamo informazioni sugli articoli che sono necessari per la loro esecuzione, questi articoli sono caratterizzati dal non avere la distinta base e rappresentano le materie prime che l'azienda deve acquisire esternamente.
Generalmente si avranno dei vincoli di propedeuticita' sulle fasi di lavorazione successive, essi rappresentano l'ordine nel quale le fasi devono svolgersi.
La presenza in azienda delle materie prime necessarie all'esecuzione di un task rappresenta per essa un vincolo, poiche' il task non può essere eseguito fintanto che' non sono disponibili i relativi materiali. La pianificazione sull'arrivo delle materie prime impone quindi un vincolo temporale sul task, per cui non risulta conveniente ritardare l'esecuzione di quel certo task in quanto si lascierebbero delle giacenze improduttive in magazzino, e non risulta neanche possibile anticipare quel task poiche' le materie prime non sono disponibili. Ci possono essere materie prime con particolari caratteristiche che impongono tali vincoli di esecuzione. Queste materie prime sono quelle per cui la giacenza in magazzino rappresenta per l'azienda un costo altissimo (es. materie preziose), per cui il task va eseguito prima possibile.

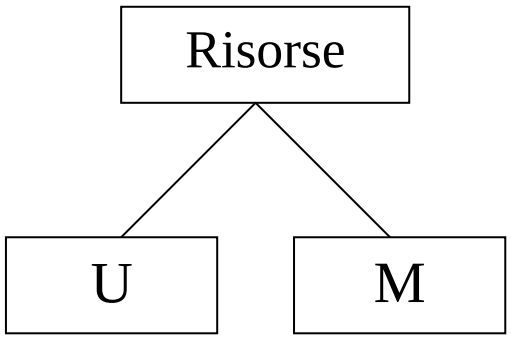


I "job" e le varie "tasks" necessarie per eseguirli, sono stabiliti sulla base delle informazioni prelevabili dal database di AMS

Job = articoli elementari da produrre
Tasks = fasi per la produzione

3.2.3 Risorse dell'azienda

Le varie fasi analizzate in precedenza con le quali si portano a termine i vari processi sono eseguite dalle risorse.
Le risorse sono caratteristiche di ogni azienda e si dividono essenzialmente in due insiemi, risorse umane e macchine. Gli elementi di entrambi gli insiemi sono caratterizzati da un parametro che risulterà fondamentale nell'esecuzione dell'algoritmo di ottimizzazione, questo parametro è il **costo per unità di tempo**.



Legenda: U = Risorse Umane
M = Macchine

Le proprietà degli elementi dei due insiemi sono estraibili dal database AMS. Tali informazioni sono fondamentali per la creazione del file di configurazione in cui vengono creati dei sottogruppi di risorse all'interno dell'azienda, ogni sottogruppo e' caratterizzato dall'essere in grado di processare ed eseguire una determinata classe di task.
Le informazioni necessarie relativamente agli elementi dell'insieme di tipo U sono:

- Costo della risorsa
- Disponibilità della risorsa
- Fasi che la risorsa può eseguire
- Modalità di esecuzione fasi, informazioni che ci dicono se l'elemento può eseguire una fase di produzione singolarmente oppure necessità della disponibilità di altri elementi dell'insieme
- Eventuale necessità di disponibilità di risorse appartenenti ad elementi dell'insieme M

Le informazioni necessarie relativamente agli elementi dell'insieme di tipo M sono simili a quelle viste sopra per gli elementi di tipo U.

3.2.4 Configurazione

Con le informazioni sulle risorse e sulle fasi si procede con l'operazione di configurazione, che consiste nel creare un file di configurazione dove le risorse dell'azienda sono elencate e classificate in gruppi di risorse (risorse di tipo U ed M) in grado di eseguire una certa fase. Ad esempio:

- **Gruppo_1_0**: u4 and m3
- **Gruppo_1_1**: u0 and m3
- **Gruppo_1_2**: u2 and u7 and m5

Tutti questi gruppi hanno la possibilità di eseguire la fase 1 (esempio fase di taglio, fase di CAD, dipende dal tipo di azienda).
Viene eseguita un'operazione simile per ogni fase.

Il primo indice rappresenta il codice della fase che le risorse elencate possono eseguire: nell'esempio sopra tutti i gruppi di risorse eseguono la fase 1. Il secondo indice rappresenta la ridondanza delle risorse: ogni singolo gruppo di risorse esegue la fase 1 e la fase 1 puo' essere eseguita nel nostro caso da tre gruppi di risorse. Il singolo gruppo di risorse e' da intendersi preso tutto insieme, ovvero la fase 1 puo' essere eseguita sulla machina m3 dall'operatore u4, oppure sulla macchina m3 dall'operatore u0, oppure sulla macchina m5 dagli operatori u2 e u7 che operano insieme.

Questa operazione non viene effettuata ogni volta che viene lanciato il programma di ottimizzazione ma ogni volta che vengono effettuate modifiche delle risorse a disposizione, ad esempio se l'azienda acquista un nuovo macchinario oppure se modifica il personale. In pratica la configurazione e' necessaria quando si hanno cambiamenti quantitativi del parco risorse aziendale.

Quando si eseguono le assegnazioni delle risorse alle fasi di lavorazione si deve controllare l'effettiva disponibilità di un dato gruppo, andando ad analizzare, con il metodo opportuno, lo stato relativo a ciascun oggetto membro del gruppo in esame. Il gruppo sarà non disponibile se soltanto uno dei membri sarà indisponibile.

3.2.5 Unità di Tempo

Il modulo Optimiser deve agire in sinergia con AMS, interagendo con esso e avendo un ugual metodo di misurazione del tempo e la stessa unita' di misura. A priori non sappiamo su che tipo di azienda opererà, quindi non possiamo sapere quale sarà l'unità di tempo che caratterizzerà le varie fasi di produzione (questo dipende dal tipo di produzione dell'azienda).
Per ovviare a questo problema si adottano delle unità di misura fittizie, che a seconda dell'ambente in cui agiremo, potranno valere minuti, secondi, ore, ecc.
Si introduce una funzione $t_v(i)$ che calcola gli istanti di tempo virtuali che rappresentano momenti in cui si hanno delle variazioni significative nell'avanzamento del processo di produzione, tipo la fine di una task. Tale funzione e' utilizzata quando si esegue l'ottimizzazione e si produce la schedula ottima.

4. L'algoritmo di Scheduling (importante)

4.1 Il modello (o struttura dati)

Il modello si astrae dal contesto più immediato del problema poiché descrive un generico problema del tipo *Resource Constrained Project Scheduling* (RCPS).

Nel modello vi sono un set di job $\mathcal{J} = \{j_1,\ldots,j_Z\}$, ognuno costituito da un certo numero di task/operazioni $j_i = \{o_{i,1},\ldots,o_{i,Li}\}$, dove L_i rappresenta il numero di task nel job i-esimo; i job sono tra loro indipendenti così come del resto lo sono i task appartenenti a job differenti; il parametro che distingue un job dall'altro è la data di consegna dello stesso.
Si può indicare con O l'insieme di tutti i task presenti nei vari job:

$O = \{o_1,\ldots,o_N\}$, dove N è dato da

$$N = \sum_{i=1}^{Z} L_i$$

La notazione utilizzata per identificare un task è biunivoca, ovvero

$$o_i \in O \Leftrightarrow o_i \in j_k\ (j_k \in \mathcal{J}) \text{ con } o_i \equiv o_{k,j} \quad , (1 \le j \le L_k)$$

È dunque possibile accedere ad un task sia tramite un numero che ne identifica la posizione nell'insieme dei task, sia mediante due indici che restituiscono rispettivamente il job di appartenenza e la posizione relativa all'interno del job stesso; tale dualismo nella rappresentazione sarà sfruttato nel prosieguo.

Il problema consiste nel processare tutti i job sulle risorse disponibili; le risorse a disposizione nel loro complesso sono identificate dall'insieme $\mathcal{R} = \{R_1,\ldots,R_R\}$ dove il generico elemento R_i rappresenta una categoria/tipologia di risorse. Ogni task può essere processato solamente su una precisa tipologia di risorsa. Nel sistema vi sono per ogni tipologia di risorsa un determinato numero (M_i) di risorse dello stesso tipo:

$$\forall R_i \in \mathcal{R},\ (1 \le i \le R), \qquad R_i = \{r_{i,1},\ldots,r_{i,Mi}\}$$

Ogni task appartenente all'insieme O dei job è caratterizzato dai seguenti parametri. La maggior parte di questi parametri sono funzionali all'algoritmo di ottimizzazione, gli altri alla sua inserzione nel contesto di un sistema di gestione.

- *id* = <*anno, cod_work, cod_subwork*>, identificativo composto da una terna di interi.
- *ind* = indice del task, pari a j quando il task è rappresentato nella forma $o_{k,j}$. Esprime la posizione relativa del task all'interno del job e serve per poter accedere facilmente ai vari task/operazioni all'interno del job stesso.
- t_{oi} = tempo stimato dell'operazione/task i-esima.
- v_1 = indice del primo task all'interno del job J_k che vincola l'esecuzione del task $o_{k,j}$.
- v_2 = indice del secondo task all'interno del job J_k che vincola l'esecuzione del task $o_{k,j}$.

- v_3 = indice del terzo task all'interno del job J_k che vincola l'esecuzione del task $o_{k,j}$.
- $R_{oi} \subseteq \mathcal{R}$ = set di possibili risorse su cui compiere il task o_i. Il set di risorse determina le macchine che possono portare a termine il task. La presenza di tale lista garantisce una notevole flessibilità al modello in quanto è possibile, ad esempio, indicare al sistema un gruppo di macchine che possono svolgere un compito rendendo cosi il sistema stesso più tollerante agli errori.
- $r_{oi} \subseteq R_{oi}$ = risorsa effettivamente assegnata al task o_i.
- $Oprec_{oi} \subseteq O$ = lista dei task che vincolano l'esecuzione di o_i. I task vincolanti appartengono necessariamente allo stesso job cui appartiene o_i, per cui se $o_i \equiv o_{k,j}$ i task vincolanti possono essere al massino tre e precisamente $o_{k,v1}$, $o_{k,v2}$, $o_{k,v3}$.
- $Osucc_{oi} \subseteq O$ = lista dei task vincolati all'esecuzione di o_i. Essi appartengono necessariamente allo stesso job di o_i, per cui se $o_i \equiv o_{k,j} \Rightarrow Osucc_{oi}$ = { $o_{k,i}$, con $1 Pi PL_k$ e $i \cong j$: $o_{k,j}$ ι $Opred_{Oki}$ }
- st_{oi} = istante di avvio del task o_i.
- end_{oi} = $st_{oi} + t_{oi}$, istante in cui termina il task o_i.
- d_{oi} = deadline o scadenza del task o_i. La *deadline* definisce l'istante limite nel quale il task deve essere concluso; tale parametro è chiaramente in relazione con la *data di consegna* (dj_k) peculiare del job cui il task appartiene. Si calcola nel seguente modo:

  $d_{oi} = dj_k - \sum_{l=j+1}^{L_k} t_{k,l}$, con $o_i \equiv o_{k,j}$.
- $tipo_{oi}$ = numero che indica univocamente la tipologia di operazione cui appartiene il task o_i. Questo parametro può assumere i valori che vanno da un minimo di 1 ad un massimo di R, essendo in stretta relazione con la tipologia di risorse presenti nel sistema. Se, ad esempio, $tipo_{oi}=2$ il task o_i può essere portato a termine solamente su una delle risorse appartenenti a R_2.

E' stata scelta la tecnica *Tabu-search* per risolvere il problema dell'ottimizzazione. La scelta è caduta su di essa principalmente per la velocità che mostra rispetto ad altre metodiche di ricerca (Simultaed Annealing e Algoritmi Genetici) e per la buona qualità delle soluzioni trovate. Altro aspetto assai interessante è la semplicità dello schema generale dell'algoritmo di base, facilmente configurabile a seconda delle esigenze. I lavori che, fra gli altri, sono stati presi in maggior considerazione per l'ideazione dell'algoritmo di scheduling sono [15], [20], [28], [26], [29], [30].
Poichè il *Tabu-search* necessita di una soluzione seme (o soluzione iniziale) per poter cominciare la ricerca, si utilizza per generare la soluzione seme un algoritmo piuttosto semplice ed efficiente che ne garantisca una di buona qualità. A tal proposito è stato rilevato da più fonti come la ricerca attuata dall'algoritmo Tabu sia positivamente influenzata dalla qualità della soluzione di partenza.

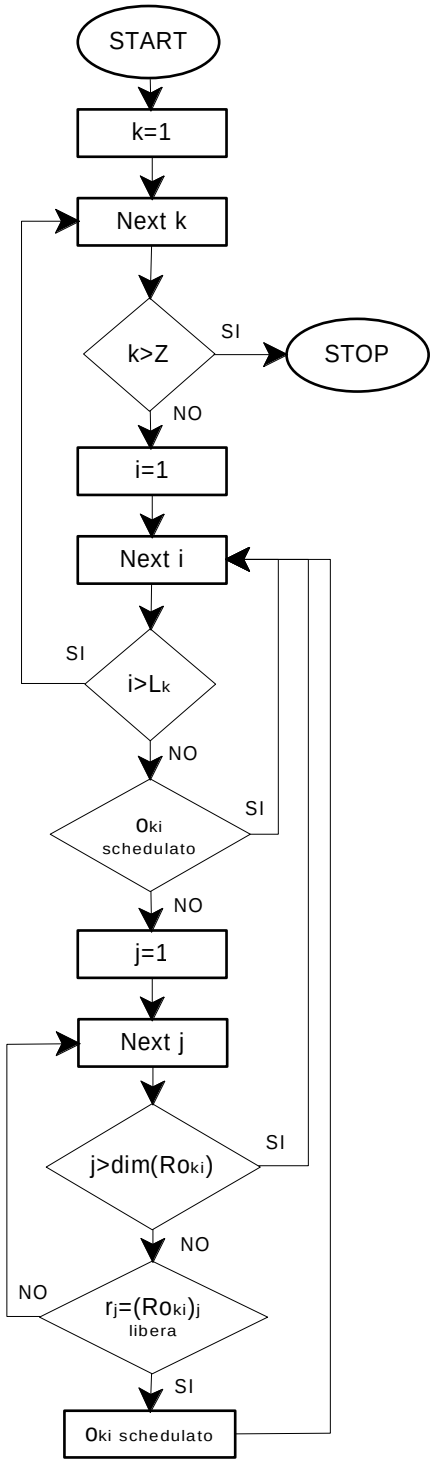
4.2 Soluzione Iniziale

Il mapping delle fasi di lavorazione elementari sulle risorse assegnate per espletarle viene fatto in automatico nella prima fase di elaborazione dell'ottimizzatore quando si crea la schedula iniziale di pianificazione della lavorazione. Nella fase successiva tale schedula verrà ottimizzata per una gestione ottimale delle risorse dell'azienda.
Le informazioni necessarie al modulo ottimizzatore per creare la schedula iniziale possono essere recuperate dal database AMS.
L'algoritmo viene invocato passandogli un vettore contenente le risorse disponibili per la pianificazione.
All'avvio vengono prelevati dal database un certo numero di job; tali job vengono riordinati dalla procedura in un vettore in base al criterio del Latest Start Time minore. Tale grandezza è definita nel seguente modo (per ogni job):

$$LST_k = dj_k - \sum_{i=1}^{L} t_{o_{k,i}} \qquad \text{con} \quad 1 \le k \le Z \quad \text{e } dj_k = \text{deadline del job k-esimo}$$

Essa rappresenta l'ultimo istante utile in cui è possibile far partire la lavorazione senza perdere la deadline (data di consegna) del job stesso. Il tempo stimato è relativo a tutti i task che compongono il job.
Nel momento in cui la procedura si avvia si pone *start* pari all'istante di partenza; *start* costituisce il primo istante virtuale. I task da eseguire vengono pianificati spostandosi in avanti nel tempo in modo virtuale. Ogni istante virtuale, a parte il primo, rappresenta la fine di uno dei task pianificati.
E' possibile schematizzare il flusso dell'algoritmo nel seguente modo:

1. Al tempo virtuale $t_v(i)$ si considerano tutti i job, con l'ordinamento dato dalla posizione nel vettore che li contiene, mappandone i task, compatibilmente con la disponibilità delle risorse e nel rispetto dei vari vincoli come mostrato nel seguente diagramma di flusso:



2. Dopo aver passato in esame tutti i job contenuti nel vettore ci si pone al tempo virtuale successivo $t_v(i+1)$, definito come il minore tra tutti i tempi di elaborazione delle fasi schedulate in precedenza e non terminate:

$$t_v(i+1) = t_v(i) + \min_{o_{ki} \subseteq O : st_{o_{ki}} \neq 0, st_{o_{ki}} + t_{o_{ki}} > t_v(i), 1 \leq i \leq L_k, 1 \leq k \leq Z} \{ st_{o_{ki}} + t_{o_{ki}} - t_v(i) \}$$

3. A $t_v(i+1)$ si procede al riordino dei job in ordine crescente in base ai loro tempi residui nel modo che segue:

$$t_{RES_k} = t_{LST_k} - t_{LOST_k}$$

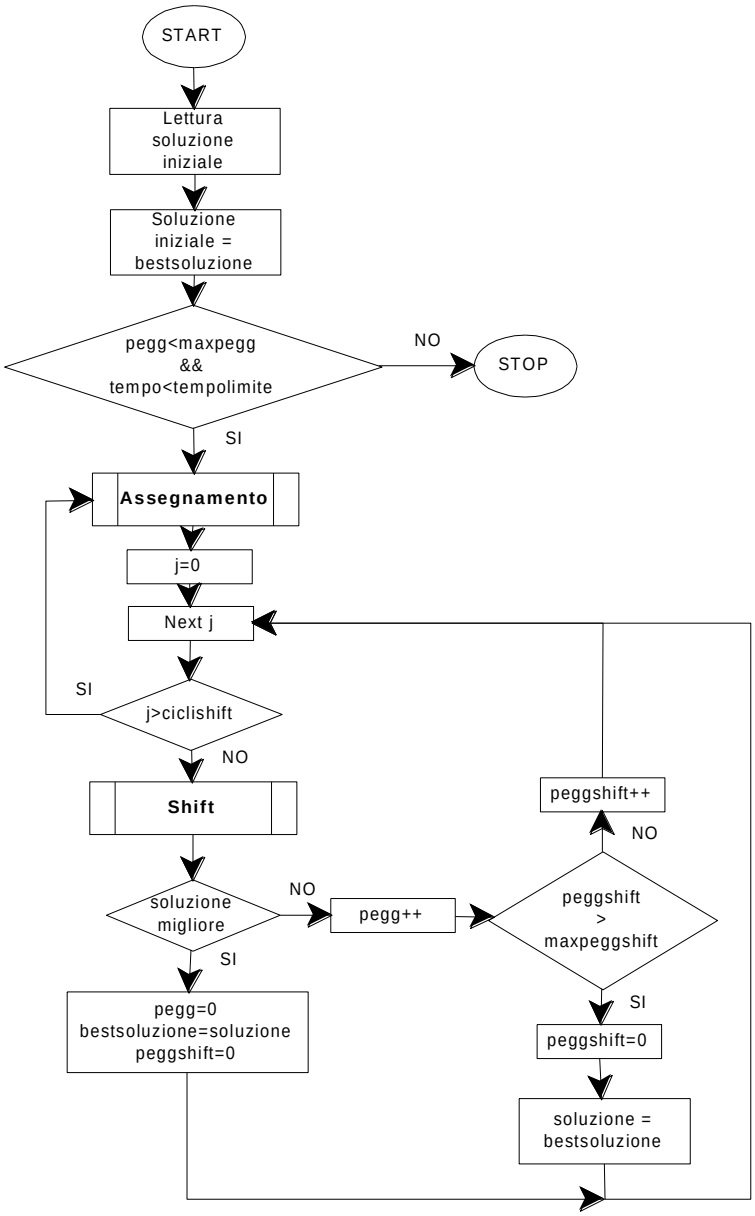
(t_{LOSTk} è la somma dei tempi in cui un job non ha avuto task schedulati). Inoltre si provvede a terminare i task che sono finiti e a rendere nuovamente libere le risorse non più allocate.

4. L'algoritmo viene terminato quando tutti i task appartenenti ai job presenti sono stati schedulati, ovvero è stato loro assegnato un istante di partenza e una risorsa sulla quale possono essere completati.

Per come è ideata la procedura di assegnazione dei task sulle risorse si vede che essa rientra nella categoria degli algoritmi *pdrs (priority dispatch rules)*; infatti è una sorta di *Earliest Due Date First* dinamico poichè ad ogni iterazione rimette in discussione le precedenze. E' da evidenziare come l'algoritmo riesca sempre a trovare una soluzione al problema di scheduling. Qualora il set di macchine sia ridotto, o più in generale ci si trovi in condizioni di carico elevato, la schedula può allungare più o meno significativamente il tempo di completamento (istante in cui termina l'ultimo task; spesso indicato con C_{max}). La soluzione potrà essere fattibile o meno a seconda che vengano rispettate le scadenze dei job che sono stati pianificati.

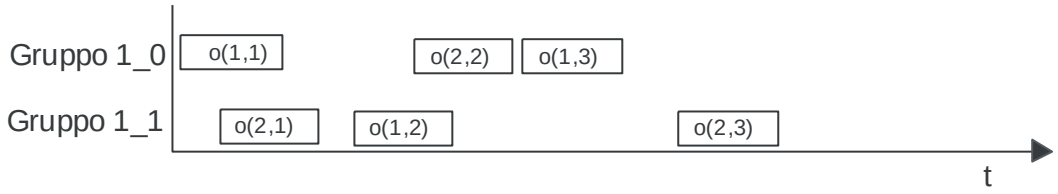
4.3 Tabu-Search

Una volta che è stata trovata una soluzione iniziale è possibile procedere con l'algoritmo Tabu. Il diagramma di flusso dell'algoritmo è il seguente:

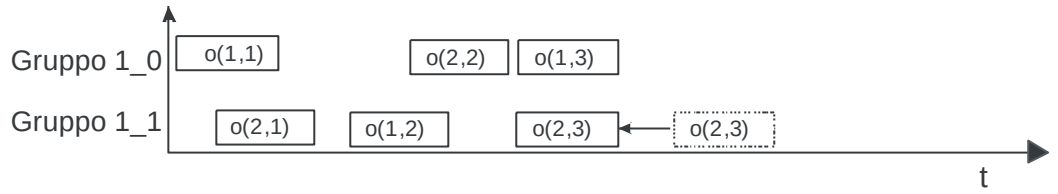


Si comincia mettendo in memoria la soluzione iniziale trovata dall'algoritmo descritto in precedenza. Questa soluzione viene, inoltre, salvata come soluzione migliore. Lo scopo prefissato è quello di migliorare sensibilmente la schedula iniziale. A tal fine la procedura prevede l'esecuzione di mosse atte a perseguire tale obiettivo. L'algoritmo cerca continuamente di trovare soluzioni migliori ma la ricerca viene interrotta qualora non si riesca a migliorare per un determinato numero di cicli di elaborazione; vi è anche una limitazione temporale utile nel caso si decida in anticipo qual è il tempo messo a disposizione dell'algoritmo. Dopo aver valutato tutte le possibili tipologie di mosse che si possono effettuare la scelta è ricaduta sulle seguenti:

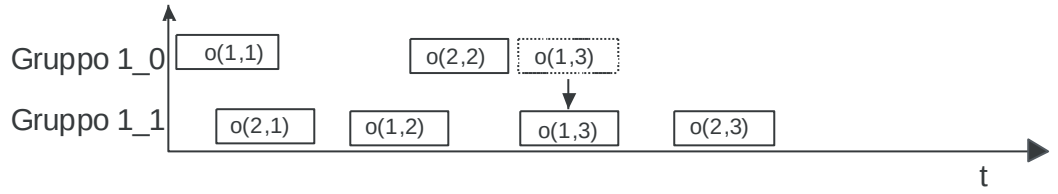
- Mossa di *shift* – uno dei task nel vettore operazioni viene anticipato o posticipato, fermo restando la sua assegnazione alla risorsa. Di seguito viene fornito un semplice esempio di una mossa di shift per un task. Si suppone che vi siano due gruppi (Gruppo_1_0 e Gruppo_1_1) e due job (J_1 e J_2). Inoltre per semplicità sono stati considerati identici i tempi di completamento di tutti i task.
  La fase *f (x, y)* riportata nelle figure seguenti si riferisce alla fase di lavorazione del job *x*, il quale e' stato suddiviso in n fasi e quella indicata e' la *y-esima*. Si fa presente che in questo esempio i job sono divisi in operazioni elementari (identificate dal secondo indice), ma tutte queste operazioni sono caratterizzate dall'essere dello stesso tipo (infatti il primo indice dei gruppi e' sempre 1). Il caso generale e' piu' complesso.



Il task che si decide di shiftare è il task $o_{2,3}$; l'entità dello shift (sia esso un anticipo od un posticipo) può assumere diversi valori a seconda dei vincoli che vi sono nella struttura della schedula. Nell'esempio, il task in questione è stato anticipato fino all'istante in cui si conclude il task $o_{2,2}$ che lo precede nel job.



- Mossa di *assegnazione* – uno dei task nel vettore operazioni viene spostato dalla risorsa alla quale è assegnato ad un'altra che figura nella lista di quelle possibili, fermo restando la propria collocazione temporale. Riprendendo il medesimo esempio si può vedere come opera una mossa di questo tipo:



Nell'esempio si è implicitamente fatto l'ipotesi che il task $o_{1,3}$ potesse essere eseguito sia su Gruppo_1_0 che su Gruppo_1_1; in termini matematici questo equivale a dire che $Ro_{1,3} = \{r_{1,1}, r_{1,2}\}$, nel caso in cui vi sia un'unica tipologia di risorse ($\mathcal{R} = \{ R_1 \}$).

Le mosse di shift hanno l'obiettivo di minimizzare il tempo di completamento della schedula, nonchè di costringere i task a rispettare le proprie scadenze, mentre **le mosse di assegnazione fanno ripartire la ricerca in condizioni differenti per esplorare regioni diverse nello spazio delle soluzioni**. Le mosse di assegnazione permettono di ottimizzare la schedula oltre che sulla base del tempo anche sulla base dei costi delle risorse.

Adottando la logica di funzionamento della Tabu-search queste mosse divengono Tabu dopo che sono state eseguite e lo rimangono per un determinato numero di cicli. A tal proposito ci sono due array, rispettivamente per le mosse di shift e quelle di allocazione, in cui è memorizzato il numero residuo di cicli per il quale la mossa rimane Tabu. Il primo array ha dimensione pari a N (numero dei task), mentre il secondo è una matrice di dimensione N*M, con $M=\max\{R_1,\ldots..,R_R\}$.

4.3.1 Funzionali di costo

Le mosse da eseguire vengono scelte in base al valore di un apposito funzionale che deve essere minimizzato. La scelta di un funzionale idoneo è assai complessa, in quanto esso determina la qualità delle soluzioni trovate. La difficoltà risiede, oltre che nel definire i termini che compongono il funzionale, anche nel dare il giusto peso ad ognuno di essi; l'aspetto più delicato è identificare un funzionale che rappresenti al meglio il tessuto connettivo della schedula e che a parità di carico computazionale riesca a fornire il maggior numero di informazioni utili per una proficua ricerca nello spazio delle soluzioni. Per quanto riguarda la struttura della schedula (nei problemi appartenenti alla categoria RCPS), essa deve, generalmente, rispettare tre tipologie di vincoli:

- vincoli sulle risorse: i task non devono essere pianificati su risorse che non hanno le caratteristiche adatte ad eseguirli (ad esempio, un task di tipo 1 su un gruppo di tipo Gruppo_2_x).
- vincoli di precedenza: task vincolati non debbono essere avviati prima che i task che li vincolano non siano terminati (ad esempio, un task 3 non deve poter essere avviato prima che il task 2 sia terminato, se esiste un vincolo di propedeuticita' fra i due task).

- vincoli di contemporaneità: task programmati sulla stessa risorsa non possono sovrapporre le loro lavorazioni (in altre parole una risorsa non può essere contemporaneamente occupata ad eseguire due o più task).
- vincolo sulla disponibilita' dei materiali: per le materie prime, la data di arrivo della materia prima rappresenta un vincolo temporale sulla schedula, sia poiche' dal momento in cui tale materia prima arriva in magazzino essa deve essere lavorata nel piu' breve tempo possibile, sia poiche' la fase non puo' essere eseguita fisicamente finche' non e' disponibile la materia prima. Tale vincolo ha particolare importanza per materie prime ad elevato costo di stoccaggio, le quali devono stare ferme in magazzino per il piu' breve tempo possibile. Togliendo tale vincolo puo' essere pianificato a priori il piano di stoccaggio delle materie prime sulla base della migliore pianificazione di produzione.

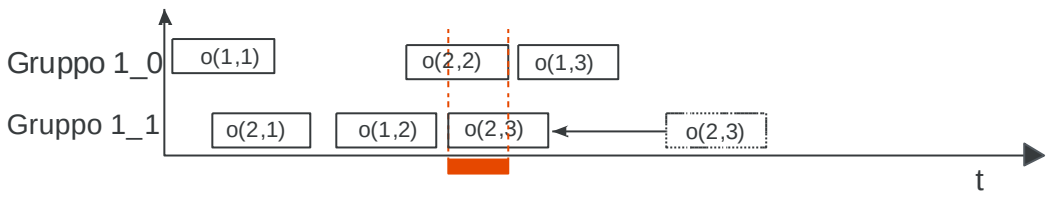
Una possibile strategia potrebbe essere quella di lasciare grande libertà alle mosse che operano sui task, permettendo loro di movimentare task indipendentemente dai vincoli suddetti e lasciando al funzionale il compito di selezionare le configurazioni più vantaggiose. La strategia in antitesi con quella precedente è, invece, quella di permettere solamente l'esecuzione di quelle mosse che non violino alcuno dei vincoli menzionati. Il vantaggio nel primo caso dovrebbe essere quello di alterare significativamente la soluzione corrente nella speranza di "scovare" soluzioni migliori; si tratta, dunque, di esplorare con grande raggio d'azione lo spazio delle soluzioni. Il maggiore svantaggio che si può presentare è inerente al fatto che si può finire in regioni dello spazio delle soluzioni assai distanti da schedule che abbiano un senso; in tali casi può essere assai difficile, se non addirittura impraticabile, tornare ad avere schedule che siano realmente delle soluzioni con la sola forza computazionale. La seconda strategia, al contrario, garantisce una sequenza di schedule genericamente fattibili con un costo computazionale relativamente basso, con il rischio però di non riuscire a trovare soluzioni che migliorino decisamente la schedula di partenza (sovente si rischia di rimanere intrappolati nei cosiddetti minimi locali).
Partendo con l'idea di applicare una strategia più aggressiva come quella descritta per prima si deve, però, scartare la possibilità di effettuare mosse che violassero i vincoli sulle risorse, in quanto verrebbe alterata in modo radicale la struttura stessa della schedula, con conseguenze assai svantaggiose in termini di performance.
Ritenendo, dunque, di mantenere la possibilità di violare i restanti vincoli e al fine di misurare l'entità delle violazioni di precedenza e di contemporaneità, si possono identificare rispettivamente i funzionali *precedence* e *allocation*.
Il funzionale *precedence*, definito nel seguente modo:

$$precedence = \frac{1}{N}\sum_{j=1}^{Z}\sum_{i=1}^{L_j}\sum_{k=1}^{Oprec_{o_{j,i}}} p_{j,i,k}$$

dove

$$p_{j,i,k} = \begin{cases} end_{o_{j,k}} - st_{o_{j,i}} & se\ end_{o_{j,k}} > st_{o_{j,i}} \\ 0 & altrimenti \end{cases}$$

è un valor medio di violazione di precedenze nella schedula; esso verifica per ogni task se nella lista dei task che lo vincolano, uno o più di quest'ultimi termina dopo lo start del task vincolato. Il termine $p_{j,i,k}$ restituisce l'entità di questa violazione. Il valore del funzionale va da 0 (caso ottimale, in quanto non vi è alcuna sorta di violazione) ad un valore maggiore di 0 che misura la violazione complessiva che si verifica in tutta la schedula.
La figura mostra un'esempio di violazione di precedenza conseguente ad un'operazione di shift:



Supponendo che il task o(2,3) sia vincolato al completamento dell'esecuzione del task o(2,2), il funzionale in questione dovrà valutare l'entità della violazione rappresentata dall'area trattegiata sottostante il grafico. L'indice appena definito, pertanto, penalizza queste configurazioni proibite della schedula
Il funzionale *allocation*, che assume la seguente forma:

$$allocation = \frac{1}{RN^2}\sum_{m=1}^{R}\frac{1}{M_m}\sum_{i=1}^{M_m}\sum_{j=1}^{N}\sum_{k=j+1}^{N} g_{j,k}\, s_{j,k}\, f_{j,k}$$

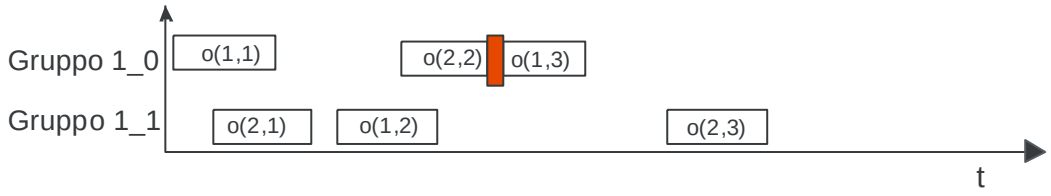
dove

$$g_{j,k} = \begin{cases} 1 & se\ r_{m,i} = r_{o_j} = r_{o_k} \\ 0 & altrimenti \end{cases}$$

$$s_{j,k} = \begin{cases} 1 & se\ st_{o_k} < end_{o_j}\ e\ end_{o_k} < st_{o_j} \\ 0 & altrimenti \end{cases}$$

$$f_{j,k} = \begin{cases} end_{o_j} - st_{o_k} & se\ st_{o_j} \le st_{o_k}\ e\ end_{o_j} \le end_{o_k} \\ end_{o_k} - st_{o_k} & se\ st_{o_j} \le st_{o_k}\ e\ end_{o_j} > end_{o_k} \\ end_{o_j} - st_{o_j} & se\ st_{o_j} > st_{o_k}\ e\ end_{o_j} \le end_{o_k} \\ end_{o_k} - st_{o_j} & se\ st_{o_j} > st_{o_k}\ e\ end_{o_j} > end_{o_k} \end{cases}$$

è invece un valor medio di violazione di contemporaneità nella schedula. Questo funzionale misura la violazione che avviene quando due task occupano (totalmente o parzialmente) la stessa risorsa nel medesimo periodo temporale; il termine $g_{j,k}$ serve ad annullare il termine della somma qualora il task j-esimo e il task k-esimo non risultassero pianificati sulla medesima risorsa $r_{m,i}$; il termine $s_{j,k}$, d'altro canto, annulla il termine della somma nel caso in cui i task j e k non siano perlomeno parzialmente sovrapposti; il termine $f_{j,k}$, invece, restituisce l'entità della sovrapposizione tra i task i e j; Il valore di tale funzionale è compreso tra 0 (nessuna allocazione contemporanea) che rappresenta la situazione ottimale ed un valore maggiore di 0 che rappresenta la violazione media. Se tutti i task avessero la stessa durata e fossero tutti sovrapposti il funzionale varrebbe esattamente la durata di un singolo task. La figura seguente illustra, riprendendo l'esempio precedente, una violazione di questo tipo:



Si suppone che si sia provveduto ad effettuare una mossa di shift (anticipo) sul task o(1,3); tale shift ha parzialmente sovrapposto i tempi di esecuzione dei task o(2,2) e o(1,3). Naturalmente tale configurazione è impossibile in quanto prevederebbe che la macchina MM0 lavori contemporaneamente due pezzi; *allocation* dà una misura della sovrapposizione indicata nella figura dall'area tratteggiata.
Lo scopo del funzionale *allocation* è, dunque, quello di penalizzare configurazioni in cui vi siano violazioni dovute a contemporanee condivisioni di risorse.

Naturalmente, oltre a identificare i termini che penalizzano le violazioni descritte in precedenza, si è dovuto costruire dei funzionali atti a minimizzare il tempo di completamento della schedula. Il funzionale più semplice da definire in questo senso è il seguente:

$$C_{\max} = \frac{\max_{o_i \subseteq O}\{end_{o_i}\} - \min_{o_i \subseteq O}\{st_{o_i}\}}{\min_{o_i \subseteq O}\{st_{o_i}\}}$$

Il funzionale C_{max} misura la lunghezza della schedula ricercando tra tutti i task quello avente il termine più lontano nel tempo (MAX(end_{oi})) e quello, invece, con il primo istante d'avvio (MIN (st_{oi})) e facendone la differenza. Il funzionale viene poi normalizzato con l'istante stesso di partenza della schedula. La minimizzazione di questo funzionale favorisce configurazioni con tempo di completamento della schedula minore.

Poichè un parametro fondamentale nella produzione industriale è il rispetto delle date di scadenza (o deadline) si è pensato di modellare tale esigenza con il seguente termine:

$$biasDeadline = \frac{1}{N}\sum_{i=1}^{N}(d_{o_i} - end_{o_i})$$

Il funzionale *biasDeadline* è il valor medio relativo all'anticipo (o al ritardo) rispetto alle scadenze delle operazioni contenute nella schedula; esso restituisce la somma degli scostamenti che ogni task ha rispetto alla propria deadline; lo scostamento viene valutato tramite la differenza tra la deadline del task e l'istante in cui esso termina. Questo funzionale può assumere valori negativi e positivi. Valori positivi del funzionale indicano una tendenza generale a rispettare le scadenze, mentre valori negativi esattamente l'opposto. Massimizzare questo termine significa premiare schedule in cui i task anticipano in maggior misura la deadline.

Il termine *biasDeadline* valuta alla stessa stregua tutti i task (siano essi in accordo con la deadline o meno) per cui se utilizzato correttamente può aiutare ad anticipare tutti i task componenti la schedula rispetto alla relativa scadenza. Per favorire, invece solamente quelli in ritardo rispetto alla deadline è stato introdotto il seguente termine:

$$delay = \frac{1}{N}\sum_{i=1}^{N} h_i$$

dove

$$h_i = \begin{cases} end_{o_i} - d_{o_i} & se\ end_{o_i} > d_{o_i} \\ 0 & altrimenti \end{cases}$$

Il funzionale *delay*, anch'esso un valor medio, favorisce i task che sono in ritardo rispetto a quelli che rientrano nella scadenza prevista. Per ottenere questo scopo il funzionale lavora in maniera analoga a *biasDeadline* misurando, però, solo gli scostamenti negativi che indicano ritardo; ciò si ottiene tramite il termine h_i, il quale restituisce solamente l'entità del ritardo o 0 qualora il task rispetti la deadline. Il funzionale viene cosi ad assumere valori da 0 (nessun task è in ritardo rispetto alla propria scadenza) a numeri positivi che indicano il ritardo medio complessivo dei task. Minimizzare questo termine significa privilegiare configurazioni che ne avvicinino a 0 il valore favorendo in tal modo il rientro dei task in ritardo nelle scadenze previste.

L'ultimo termine da prendere in considerazione è il seguente:

$$\mathrm{var}\,Totale = \frac{1}{RN^2}\sum_{m=1}^{R}\frac{1}{M_m}\sum_{i=1}^{M_m}\left(\sum_{j=1}^{N} c_j t_{o_j} - \sum_{k=1}^{N} b_k t_{o_k}\right)^2$$

dove

$$c_j = \begin{cases} 1 & se\ r_{m,i} = r_{o_j} \\ 0 & altrimenti \end{cases}$$

$$b_k = \begin{cases} 1 & se\ tipo_{o_k} = m \\ 0 & altrimenti \end{cases}$$

Il funzionale *varTotale* di per sè non ha benefici immediati sulla qualità della schedula; esso costituisce una sorta di valor medio di carico. Il suo contributo consiste nell'uniformare il carico di lavoro sulle risorse presenti nel sistema. Opera nel seguente modo: vengono esaminate tutte le risorse appartenenti alle varie tipologie che sono presenti nel sistema e per ognuna di esse si valuta lo scostamento del carico rispetto a quello che mediamente si presenta sulle risorse appartenenti alla stessa tipologia. Infatti il termine $(\sum_{j=1}^{N} c_j t_{o_j} - \sum_{k=1}^{N} b_k t_{o_k})^2$ rappresenta lo scarto quadratico medio tra la somma dei tempi stimati dei task pianificati sulla risorsa $r_{m,i}$ e la media dell'analoga quantità valutata su tutte le risorse di tipologia m. È plausibile, a tal riguardo, che configurazioni meno congestionate possano condurre con l'ausilio di mosse di spostamento (shift) temporale a soluzioni migliori. Rendendo sempre più piccolo questo termine si tende ad uniformare il carico sulle risorse. L'utilizzo di questo funzionale e' opportuno nel caso che i gruppi di risorse abbiano un costo simile tra di loro.

Nel caso invece in cui i gruppi di risorse abbiano costi sensibilmente diversi conviene sfruttare maggiormente, dal punto di vista economico, quei gruppi di risorse che presentano un costo inferiore. In tal caso il funzionale *varTotale* non viene considerato nel funzionale totale di costo e si utilizza invece il funzionale *ResEffort*.
Dato il costo di ogni gruppo di risorse per unita' di tempo, il funzionale *ResEffort* e' dato dalla somma del costo di ogni gruppo per il tempo in cui quel gruppo e' impiegato.

$$ResEffort = \sum_{j,k} c_{j,k} * T_{j,k}$$

dove:

- $c_{j,k}$ = costo per unità di tempo del Gruppo_ j _k
- $T_{j,k}$ = tempo totale di utilizzo in unita' di tempo del Gruppo_ j _k

4.3.2 Scelta del funzionale di costo da minimizzare

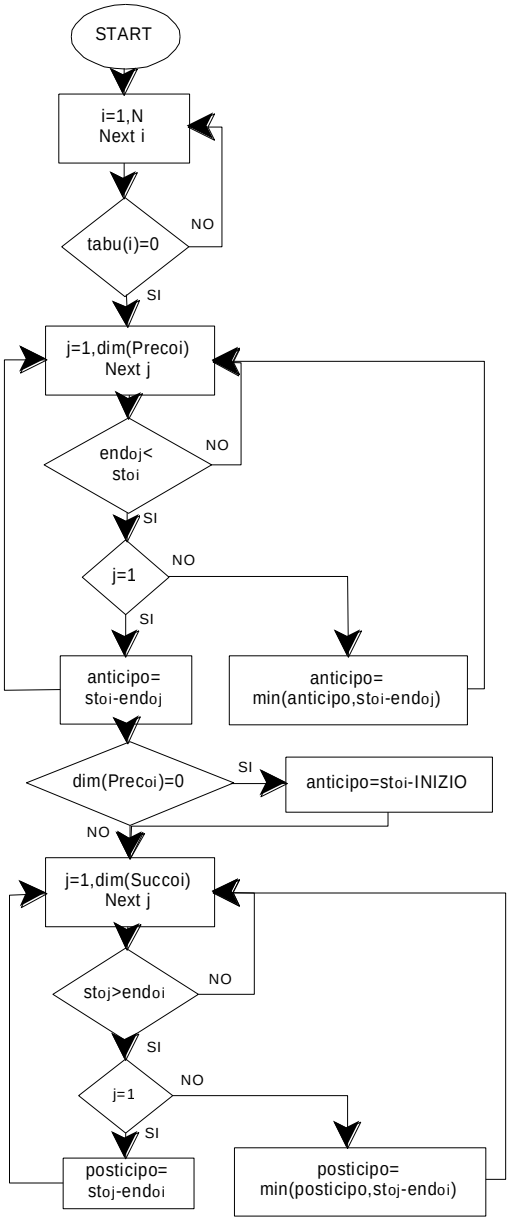
Per stabilire quali termini tra quelli presentati facciano parte o meno del funzionale F da minimizzare si deve valutare il comportamento dell'algoritmo con i termini considerati singolarmente o in combinazione tra loro. Si riscontra, permettendo che la mosse possano violare i vincoli di precedenza, che il carico computazionale e' eccessivo. Ogni termine del funzionale F ha un costo in termini computazionali: non si considerano mosse che portano a violazioni di precedenza, risparmiando, al contempo, il tempo speso per la valutazione di *precedence*.
Per quanto riguarda, invece, le violazioni di contemporaneità si lascia questa caratteristica in quanto il termine del funzionale *allocation* generalmente riesce a risolvere tali incongruenze.
Per quanto riguarda i restanti termini nessuno di essi, da solo, riesce a far sì che si ottengano soluzioni soddisfacenti. C_{max} favorisce solamente le mosse che anticipano i task che riducono il tempo di completamento della schedula (cioè solo gli ultimi); *biasDeadline* e *delay* tendono ad anticipare tutti i task ma non risolvono le violazioni di contemporanea allocazione di task; *allocation* non si preoccupa in alcun modo di anticipare l'esecuzione dei task come del resto *varTotale*.
I fattori assolutamente necessari sono *allocation* e perlomeno uno tra *biasDeadline*, *delay* e C_{max}; senza il primo, infatti, si ottengono schedule con molti task sovrapposti, senza uno di quelli del secondo gruppo, invece, non si riesce a ridurre la schedula.
Inoltre, per quanto riguarda questi ultimi, il funzionale C_{max} e' altamente correlato con il funzionale *biasDeadline*: variazioni positive o negative di quest'ultimo si ripercuotono con lo stesso segno sull'altro. Si puo' rinunciare al contributo di C_{max}, in quanto esso agisce solamente sui task all'inizio ed alla fine della schedula non favorendo gli altri.

Riguardo al termine *varTotale*, la sua presenza permette di distribuire uniformemente i carichi di lavoro sulle risorse presenti. In alcuni problemi questo aspetto si può rivelare vantaggioso, in altri, invece, va a scapito della ricerca del maggior anticipo rispetto alle scadenze dei task e della riduzione del tempo di completamento della schedula.
Il termine del funzionale *ResEfforts* permette di ottimizzare la schedula non solo sulla base del tempo, ma anche del costo delle varie risorse.
I valori dei parametri effettivi di pesatura e la calibrazione di ogni termine del funzionale nel funzionale di costo totale, cosi' come la valutazione dell'impatto di ognuno sull'ottimizzazione sara' il risultato di una operazione di calibratura da farsi in sede di implementazione e applicazione dell'algoritmo di ottimizzazione nei casi d'uso previsti.
Quanto detto viene riassunto nella seguente tabella:

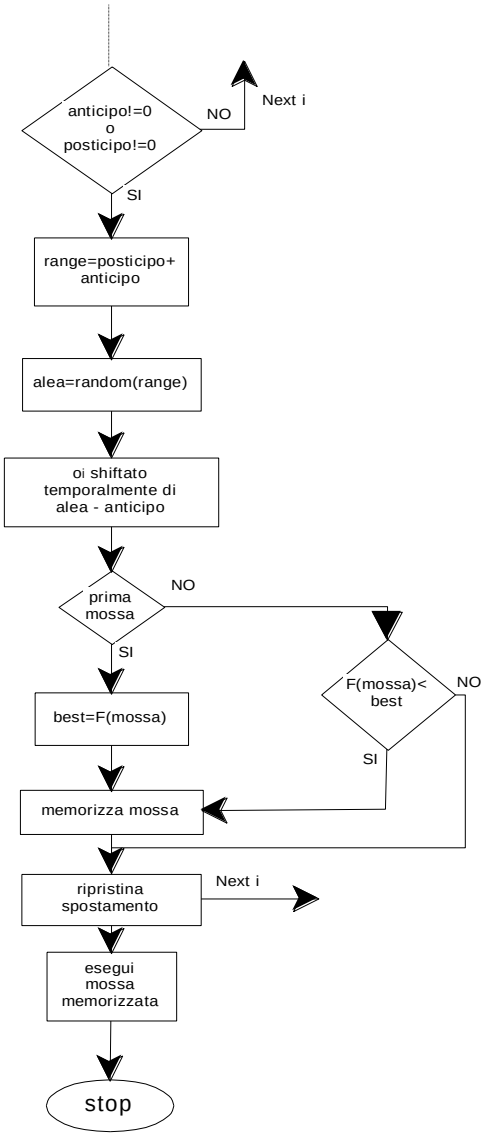
	Anticipa scadenze	**Riduce schedula**	**Risolve violazioni**	**Minimizza costi delle risorse**	**Uniforma carico**
biasDeadline	SI	SI	NO	NO	NO
delay	SI per i task in ritardo	SI se task in ritardo	NO	NO	NO
C_{max}	SI ma solo dell'ultimo task	SI	NO	NO	NO
allocation	NO	NO	SI	NO	NO
varTotale	NO	NO	NO	NO	SI
ResEffort	NO	NO	NO	SI	NO

4.3.3 Applicazione di una mossa di shift

L'algoritmo, dunque, partendo dalla configurazione iniziale proposta effettua un numero di mosse di shift pari al valore di un parametro presente nell'algoritmo (*ciclishift)*. Alla fine delle mosse di shift effettua una mossa di assegnazione utile per esplorare una regione differente dello spazio delle soluzioni. La procedura si ripete fino a quando non viene soddisfatto il criterio di arresto.
Una singola mossa di shift avviene come segue: si considerano uno per uno tutti i task presenti; per ogni task si trova il massimo anticipo effettuabile. Questo anticipo, al fine di non violare i vincoli di precedenza, viene calcolato prendendo tutti i task che vincolano quello in oggetto e valutando la distanza che separa la fine del task vincolante con quello vincolato. Il valore più piccolo tra quelli trovati sarà il massimo anticipo praticabile. D'altro canto non è possibile solamente pensare di anticipare i task pertanto viene calcolato anche il massimo posticipo praticabile in modo simile a quanto fatto in precedenza considerando questa volta i task che sono vincolati all'esecuzione di quello in esame e valutando la distanza tra la fine del task in esame e l'inizio di quelli vincolati a quest'ultimo. Il valore minore tra quelli trovati costituisce il posticipo. Il motivo per il quale si effettuano anche dei posticipi, che per loro natura tendono ad aumentare il tempo di completamento della schedula o comunque a ritardare l'avvio delle operazioni, è che in alcune situazioni si rende necessario creare dello spazio nella schedula per potervi inserire uno o più task. La sequenza di operazioni seguita è rilevabile dal seguente diagramma di flusso:

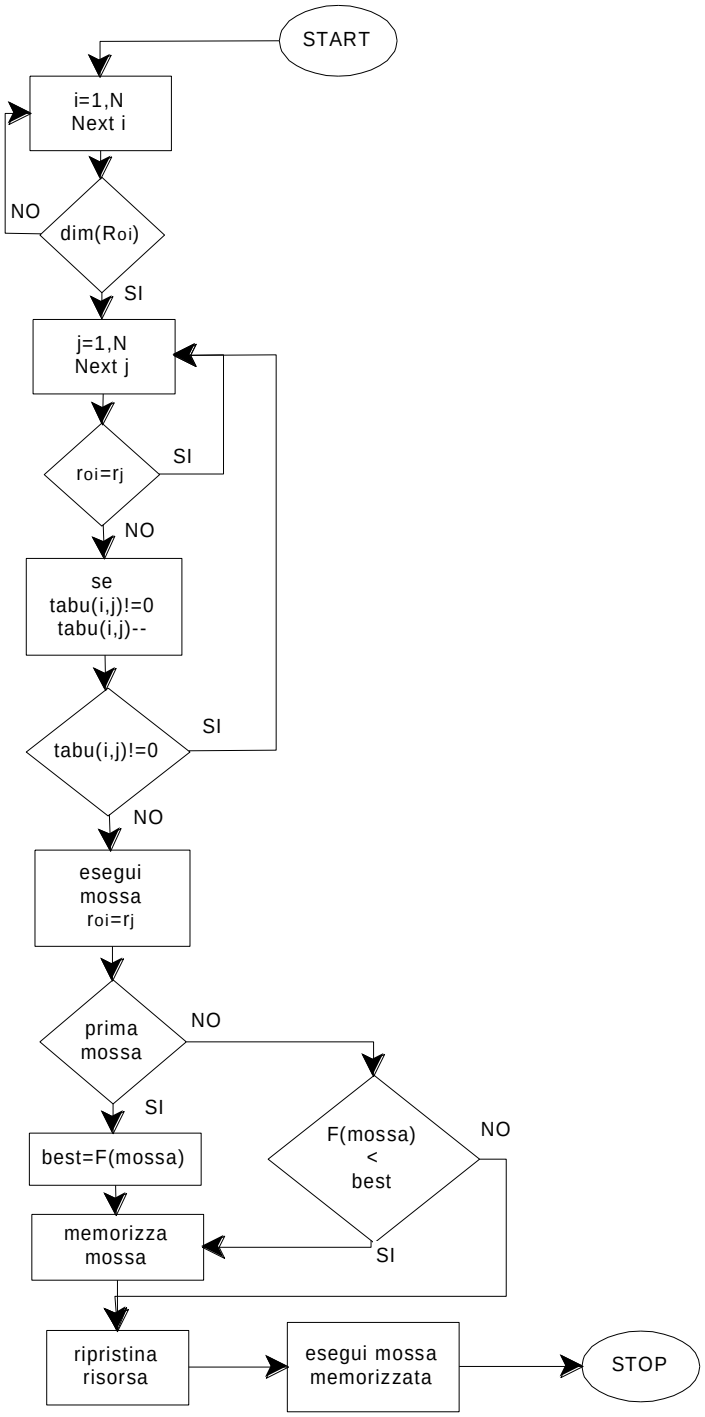


vi è da notare come nel caso in cui un task non abbia operazioni che ne vincolano l'esecuzione il relativo valore di anticipo sia pari a st_{oi} – INIZIO, questo significa che esso può essere anticipato fino all'istante di avvio dell'intera lavorazione.
Arrivati a tal punto si calcola aleatoriamente un valore compreso tra anticipo e posticipo e con tale valore si procede alla mossa di shift (tale mossa può, quindi, essere un anticipo o un posticipo). Dopo aver effettuato la mossa se ne valuta la bontà mediante il funzionale di costo precedentemente definito e si riposiziona nuovamente il task nella posizione che occupava prima di essere shiftato, memorizzando, al contempo, il valore assunto dal funzionale e l'entità dello shift.
Tutto ciò viene ripetuto per ognuno dei task e quello la cui mossa ha raggiunto la valutazione migliore viene scelto per essere shiftato della quantità memorizzata, come risulta dal relativo diagramma di flusso:



4.3.4 Applicazione di una mossa di assegnazione

La mossa di assegnazione funziona in modo molto simile esaminando tutti i task che sono presenti. Poichè si vuole assegnare un task ad un gruppo di risorse differente da quello sul quale si trova correntemente si esamina la lista dei gruppi di risorse propria del task; in questa lista vi sono i riferimenti ai gruppi di risorse su cui può essere allocato il task. Si assegna il task ad ognuno dei gruppi di risorse indicati eccetto quello corrente e si valuta il valore che il funzionale assume dopo la mossa memorizzandolo assieme alla mossa stessa e al gruppo di risorse su cui è avvenuta l'assegnazione. Dopo aver fatto ciò si ritorna alla condizione di partenza precedente la mossa ripetendo la procedura per tutti i task. Alla fine si procede all'assegnazione del task che ha ottenuto il punteggio migliore. Il flow-chart che segue riassume i concetti precedentemente esposti:



In ambedue le procedure una mossa viene valutata solamente se non è Tabu. Se viene scelta allora diviene Tabu e non viene eseguita per un numero di iterazioni pari ad un parametro impostabile.
A tal proposito vi è da notare che ogni volta che viene eseguita una mossa di shift o di assegnazione gli array deputati alla gestione delle mosse Tabu decrementano di una unità il valore che tiene conto dei cicli residui.
Per velocizzare la ricerca è possibile selezionare la percentuale di task che vengono esaminati ad ogni ciclo di shift o di allocazione. Sebbene la velocità di ricerca aumenti, al di sotto di un certo valore di percentuale di selezione dei task la qualità delle soluzioni trovate decresce progressivamente in modo sensibile.

L'algoritmo effettua continuamente mosse di shift e di assegnamento tenendo in memoria la soluzione migliore che è stata trovata nel corso della ricerca fino al passo corrente. Al fine di esplorare regioni differenti dello spazio delle soluzioni ed in particolare per cercare di non rimanere intrappolati in minimi locali l'algoritmo consente di accettare durante la ricerca soluzioni che sono peggiori di quelle precedente. Anche questa caratteristica è regolata dal valore di un apposito parametro (*peggioramentiShift)*; si accettano un numero consecutivo di soluzioni peggiori di quella

migliore, al momento incontrata, fino a che tale numero non è pari al valore del parametro. Dopo che tale valore è stato superato si riparte dalla migliore soluzione che è stata trovata.
Il criterio di arresto termina l'esplorazione dello spazio delle soluzioni quando dopo un numero di cicli pari al parametro *maxpeggioramenti* non si riesce a migliorare la soluzione. Oltre a ciò è presente un criterio di arresto relativo al tempo massimo concesso per la ricerca. La ricerca, dunque, termina quando una delle due condizioni viene soddisfatta.

5 Bibliografia

[1] M.-C. Portmann, A. Vignier, D. Dardilhac, D. Dezalay - Branch and boud crossed with GA to solve hybrid flowshops - European Journal of Operational Research 107, 1998, pp.389-400
[2] G.I. Adamopulous, C.P. Pappis - Scheduling under a common due-date on parallel unrelated machines - European Journal of Operational Research 105, 1998, pp.494-501
[3] J. Herrmann, J.-M. Proth, N.Sauer - Heuristics, for unrelated machine scheduling with precedence constraints - European Journal of Operational Research 102, 1997, pp.528-537
[4] T.C. E. Cheng, M. Y. Kovalyov - Complexity of parallel machine scheduling with processing-plus-wait due date to minimize maximum absolute lateness - European Journal of Operational Research 114, 1999, pp.403-410
[5] A.S. Jain, S. Meeran - Deterministic job-shop scheduling: Past, present and future - European Journal of Operational Research 113, 1999, pp.390-434
[6] P.B. Luth, D. Chen, L.S. Thakur - An Effective Approach for Job-Shop Scheduling with Uncertain Processing requirements - IEEE Transaction on Robotics and Automation, vol. 15, No. 2, April 1999
[7] J. Liu, B.L. MacCarthy - A global MILP model for FMS scheduling - European Journal of Operational Research 100, 1997, pp.441-453
[8] K. Shanker, B. K. Modi - A branch and bound based heuristic for multi-product resource constrained scheduling problem in FMS environment - European Journal of Operational Research 113, 1999, pp.80-90
[9] G. Pistolesi - Genetica computazionale - Byte Italia, Luglio/Agosto 1999, pp.60-65
[10] J. H. Holland - Algoritmi genetici - Le Scienze, N° 289, Settembre 1992
[11] C. Bierwirth - A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms - Baltzer Journal
[12] D. C. Mattfeld, C. Bierwirth - A Search Space Analysis of the Job Shop Scheduling Problem - Baltzer Journal, aprile 1996
[13] C. Bierwirth, H. Kopfer, D. C. Mattfeld, I. Rixen - Genetic Algorithm based Scheduling in a Dynamic Manufactoring Environment - Baltzer Journal
[14] M. Tucci - Algoritmi Genetici ed evoluzione naturale, robustezza dei metodi di ricerca tradizionali, … - Dispense del Dipartimento di Energetica Sergio Stecco, Università di Firenze
[15] F. Glover, E. Taillard, D. de Werra - A user's guide to Tabu search - Annals of Operations Research 41, 1993, pp.3-28
[16] F. Dammeyer and S. Voβ - Dynamic Tabu list management using the reverse elimination method - Annals of Operations Research 41, 1993, pp.31-46
[17] B. Fox - Integrating and accelerating Tabu search, simulated annealing, and genetic algorithms - Annals of Operations Research 41, 1993, pp.47-67
[18] J. P. Kelly, B. L. Golden, A. A. Assad - Large-scale controlled rounding using Tabu search with strategic oscillation - Annals of Operations Research 41, 1993, pp.69-84
[19] P. Moscato - An introdution to population approaches for optimization and hierarchical objective functions: A discussion on the role of Tabu search - Annals of Operations Research 41, 1993, pp.85-121
[20] P. Brandimarte - Routing and scheduling in a flexible job-shop by Tabu search - Annals of Operations Research 41, 1993, pp.157-183
[21] M.Charest, J. Ferland - Preventive maintenance scheduling of power generating units - Annals of Operations Research 41, 1993, pp.185-206
[22] D. L. Woodruff - E. Zemel - Hashing vectors for Tabu search - Annals of Operations Research 41, 1993, pp.123-137
[23] E. L. Mooney, R. L. Rardin - Tabu search for a class of scheduling problems - Annals of Operations Research 41, 1993, pp.185-206

[24] B. Srivastava, W.-H. Chen - Part type selection problem in flexible manufactoring systems: Tabu search algorithms - Annals of Operations Research 41, 1993, pp.279-297
[25] M. Dell'Amico, M. Trubian - Applying Tabu search to the job shop scheduling problem - Politecnico di Milano, Italia, 1993
[26] J. B. Chambers, J. W. Barnes - Reactive Search for Flexible Job Shop Scheduling - New Tabu Search Results for the Flexible Job Shop Problem, INFORMS, Dallas, October 1997
[27] M.G.A. Verhoeven - Tabu search for resource-constrained scheduling - European Journal of Operational Research 106, 1998, pp.266-276
[28] V. Valls, M. A. Perez, M. S. Quintanilla - A Tabu search approach to machine scheduling - European Journal of Operational Research 106, 1998, pp.277-300
[29] M. Tucci, R. Rinaldi - From theory to application: Tabu search in textile production scheduling - Pre-print version, Dipartimento di Energetica Sergio Stecco, Università di Firenze, 1999
[30] Glover, Laguna - Tabu Search - Kluwer, Norwell MA, 1997
[31] J. Józefowska, M. Mika, R. Rózycki, G. Waligóra, J, Weglarz - Local search metaheurics for discrete-continuous scheduling problems - European Journal of Operational Research 107, 1998, pp.354-370
[32] M. Mitchell - Introduzione agli algoritmi genetici - Apogeo, 1999
[33] F. Shoen - Teoria e metodi di ottimizzazione lineare - La Nuova Italia Scientifica, 1991
[34] F. Arnaldi, M. Bignami - La schedulazione delle attività di fabbricazione in un'azienda machinery - Meccanica e Macchine, 1998
[35] M. Garantini - Il futuro dell'automazione di processo - automazione e Strumentazione, gennaio 1999
[36] G. Zampelli - Tra l'ERP e il controllo, c'è di mezzo il MES - Automazione Industriale, Marzo 1999
[37] M. Flego - L'impiego del controllo numerico nella produzione meccanica - Franco Angeli ed., 1975
[38] T.C.E. Cheng, Z.L. Cheng - Parallel-machine scheduling problems with earliness and tardiness penalties, A state of the art review of parallel machinesscheduling research - European Journal of Operational Research 47, 271-291.
[39] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi - Optimization by simulated anealing - Science, 220, 4598, 1983
[40] Y.C. Ho - A new paradigm for stochastic optimization and parallel simulation, Discrete Event Systems, Manufacturing Systems, Communication Networks - New York: Springer-Verlag, 1995, pp59-72.
[41] J. Liu, B.L. MacCarthy - The classification of FMS scheduling problems - International Journal of Production Research, 34, 647-656, 1996.
[42] B.L. MacCarthy, J. Liu, Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling - International Journal of Production Research, 31 (1), 59-79, 1993
[43] E. Amaldi, E Mayoraz, D. de Werra - Discrete optimization problems in neural network design - DMA preprint, EPF Lausanne, Switzerland (August 1990).
[44] J.H. Holland - Adaptation in Natural and Artificial Systems - University of Michigan Press, Ann Arbor, 1975
[45] P. Moscato - On evolution, search, optimization genetic algorithms and martial arts: Towards memetic algorithms - CalTech Concurrent Computation Program Report 826, CalTech, Pasadena, CA, 1989.
[46] P.J.M van Laarhoven, E.H.L. Aarts - Simulated annealing: Theory and Applications - Reidel, 1987.
[47] P.J.M van Laarhoven, E.H.L. Aarts, J.K. Lenstra - Job shop scheduling by simulated annealing - Report OS-R8809, CWI, Amsterdam 1988.
[48] S. Bittanti - Identificazione dei modelli e controllo adattativo - Pitagora Editrice, Bologna, 1992.
[49] S. Bittanti - Teoria della predizione e del filtraggio - Pitagora Editrice, Bologna, 1993
[50] R.L. Daniels, J.B. Mazzola - A Tabu search heuristic for the flexible-resource flow shop scheduling problem - Annals of Operational Research, 41, pp 207-230, 1993.
[51] M.Laguna, J.W. Barnes, F. Glover - Scheduling jobs with linear delay penalties and sequence dependent setup costs and times using Tabu-searc - Appl. Int. - pp. 253-260 - 1992.
[52] M. Widner - Job shop scheduling with tooling constraints: a Tabu search approach, Journal of Operational Research Society (42) - pp. 75-82 - 1991.
[53] D. Knuth - The Art of Computer Programming - Addison-Wesley, Reading, MA - 1983.
[54] W.H. Chen, B. Srivastava - A simulated annealing algorithm for production planning in flexible manufacturing systems - Working paper, Department of Management and Systems, Washington State University - June 1991.
[55] I.J. Chen, C.H. Chung - Effects of loading and routing decisions on performance of flexible manufacturing systems - Int. J. Prod. Res. 29 (1991) 2209-2225.
[56] F. Della Croce, R. Tadei, G. Volta - A genetic algorithm for the job shop problem - D.A.I. Politecnico di Torino 1992.
[57] J.L. McClelland, D.E. Rumelhart, PDP Research Group - Parallel Distributed Processing, Volume 1: Foundations - MIT Press, 1986.
[58] J. Jòzefowska, G. Waligòra, J. Weglarz - A Tabu search algorithm for some discrete continous scheduling problems - V.J. Rayward-Smith (Ed.), Modern Heuristics Search Methods, Wiley, pp.169-182 - New York, 1996.

[59] E. Balas, G. Lancia, P. Serafini, A. Vazacopoulos - Job shop scheduling with deadlines - Journal of Combinatorial Optimization 1 (4) 324-353 - 1998.
[60] E. Balas, A. Vazacopoulos - Guided local search with shifting bottleneck for job shop scheduling - Management Science 44 (2), 262-275 - 1998.
[61] F. Glover - Tabu search fundamentals and uses. Working paper - College of Business and Administration and Graduated School of Business Administration, University of Colorado, Boulder, Colorado, USA - 1995.
[62] J.R. Jackson - Scheduling a production line to minimise maximum tardiness - Research Report 43, Management Science Research Projects, University of California, Los Angeles, USA - 1955.
[63] A.S. Jain, S. Meeran - Job shop scheduling using neural networks - International Journal of Production Research 36 (5), 1249-1272.
[64] S.M. Johnson - Optimal two and three stage production schedules with set-up times included - Naval Research Logistics Quarterly 1, 61-68 - 1954.
[65] M. Laguna, J.W. Barnes, F. Glover - Intelligent scheduling with Tabu search: an application to jobs with linear delay penalties and sequence dependent set up costs and times - Journal of Applied Intelligence 3, 159-172 - 1993.
[66] E .Nowicki, C. Smutnicki - A fast Tabu search algorithm for the job shop problem - Management Science 42 (6), 797-813 - 1996.
[67] E. Taillard - Parallel Tabu search for the job shop scheduling problem - ORSA Journal of Coumputing 16 (2), 108-117 - 1994.
[68] K. Ramamritham - Allocation and Scheduling of Precedence-Related periodic tasks - IEEE Transaction on Parallel and Distribuited Systems, Vol 6, n°4, Aprile 1995 pp. 412-420
[69] N. Seinosuke, K. Hironori - Pratical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing - IEEE Transaction on Computers, Vol. c-33, n°11, Novembre 1984 pp. 1023-1029
[70] P. Y.R. Ma, E.Y.S. Lee, M. Tsuchia - A task allocation model for distributed computing systems - IEEE Transaction Computing, Vol c-31, 1982
[71] S. C. Graves, A. H. G. Rinnoy-Can, P. H. Zipkin - Logistics of production and inventory - North Holland 1995
[72] C. Bierwirth, H. Kopfer, D. C. Mattfeld, I. rixen - Genetic Algorithm based Scheduling in a Dynamic Manufactoring Environment, Baltzer Journal
[73] A. Diwan, D. Tarditi, E. Moss - Memory system performance of programs with intensive heap allocation - ACM Transaction on Computer System, Vol. 13 pp. 244-273, Agosto 1995
[74] E. Falkenauer, S. Bouffoix - A genetic algorithm for Job Shop - Prooceedings of the 1991 IEEE International Conference on Robotics and Automation, 1991, pp. 824-829.
[75] F. Glover - Tabu Search - ORSA Journal on Computing 1,3 (1989) e 2,1 (1990).
[76] H. Matsuo, Su C. Juck, R.S. Sullivan - A controlled search simulated annealing method for general job shop scheduling problem - University of Texas, Austin, 1988.
[77] M. Mocali, T. Paoletti - Programmazione operativa della produzione: applicazione di nuove tecniche di ricerca operativa ad un caso reale - Tesi di Laurea - Università degli studi di Firenze (Dipartimento di Energetica), Dicembre 1993.