

UNIVERSITÀ DEGLI STUDI DI MILANO
Facoltà di Scienze Matematiche, Fisiche e Naturali
Dipartimento di Tecnologie dell'Informazione



Algoritmi costruttivi per un problema di sequenziamento ottimo

Relatore: Prof. Giovanni RIGHINI

Correlatore: Prof. Roberto CORDONE

Tesi di Laurea di:
Devis G. BONIZZONI
Matr. n° 617262

Anno Accademico 2002/2003

*Ai miei genitori ad Alessia
e a Gloria*

Ringraziamenti

*“Dubitare di se stessi è il
primo segno dell'intelligenza...”
Ugo Ojetti*

Mi sento in dovere di ringraziare tutte le persone che mi hanno sostenuto durante l'implementazione e la stesura di questa tesi di laurea.

Mi rendo conto che elencarle tutte sarebbe impossibile, primo perché non saprei quando terminare, secondo perché altrimenti invece che leggere il resto della pubblicazione converrebbe fare tutt'altro.

Tuttavia non posso esimermi dal nominare alcune persone che si sono fondamentali per il mio progetto.

- Il professor Giovanni Righini, il professor Roberto Cordone, ed il mio compagno di progetto Andrea, ai quali devo un enorme ringraziamento per i loro consigli fondamentali che mi hanno permesso di risolvere tutte le situazioni critiche, nelle quali sono incorso, nel migliore dei modi.
- Tutta la mia famiglia che mi ha sempre sostenuto ed aiutato a superare ogni difficoltà in questi anni di studio.
- La mia ragazza che ha sopportato le mie lamentele, i miei periodi neri, donandomi spesso consigli utili e soprattutto spendendo ore del suo tempo a confortarmi.
- Tutti quanti i miei amici, e tutti coloro che non ho nominato, sappiate che non l'ho fatto solo perché siete troppi.

A tutti mi sento di dire GRAZIE per tutto quello che volontariamente o involontariamente hanno fatto per consentirmi di finire questo lavoro.

Indice

1. Il contesto	1
1.1 Processi di pianificazione e schedulazione	2
1.1.1 Requisiti della linea di assemblaggio	3
1.1.2 Ratio constraints ad alta priorità facili e difficili da soddisfare	4
1.1.3 Requisiti della verniciatura	5
1.1.5 Ruolo del giorno di produzione precedente.	6
1.2 Problema da risolvere	6
1.2.1 Ottimizzazione multi-obiettivo con massima priorità ai ratio constraints.	7
1.2.2 Ottimizzazione multi-obiettivo con massima priorità ai cambi di colore	8
1.3 Dettagli sul calcolo del numero di violazioni dei ratio constraints	9
1.4 Dettagli sul calcolo del numero di cambi di colore	12
2. Il concorso	13
2.1 Linguaggi ammessi ed ambiente di test.	13
2.2 Organizzazione dei files ed interpretazione della linea di comando	14
2.3 Procedure di valutazione e classificazione.	15
2.3.1 Il programma di valutazione	16
2.3.2 Metodo di valutazione di uno scenario	16
2.3.3 Valutazione globale di un test set	18
2.4 Descrizione dei test set	20
2.4.1 Contenuto di un test set	20

2.5 Formato dei files	21
2.5.1 File della limitazione delle sequenze di auto con lo stesso colore .	21
2.5.2 File degli obiettivi di ottimizzazione	22
2.5.3 File dei ratio constraint	23
2.5.4 File dei veicoli	23
2.5.5 File delle soluzioni	26
3. Strutture generate	27
3.1 Diagramma di flusso globale	28
3.2 Inizializzazione e caricamento dati.	30
3.2.1 Strutture dati.	31
4. La fase costruttiva	35
4.1 Generazione della sequenza finale.	37
4.2 Inizializzazione delle strutture e calcolo delle densità dei vincoli.	37
4.3 Generazione della soluzione per righe	38
4.3.1 Generazione della soluzione basata sul minimo numero di violazioni sui ratio constraints ad alta priorità	39
4.3.2 Generazione della soluzione basata sul minimo numero di cambi di colore	47
4.4 Generazione della soluzione per colonne.	53
4.4.1 Generazione della soluzione basata sul minimo numero di violazioni sui ratio constraints ad alta priorità	53
4.4.2 Generazione della soluzione basata sul minimo numero di cambi di colore.	58
4.5 Confronto tra generazione per righe e per colonne	60
5. Ricerca locale	65
6. Risultati	67
6.1 Risultati del test set A	68
6.2 Gestione del limite di tempo	70

6.3 Risultati del profiling	71
7. Possibili miglioramenti	75
7.1 Migliorare la qualità dei risultati	75
7.2 Incrementare la velocità di esecuzione.	76
7.3 Miglioramenti generali	76
Appendice	V
A. Indice delle immagini.	V
Bibliografia	77

Indice delle Immagini

Diagrammi

Diagramma 1: Funzionamento generale	28
Diagramma 2: Inizializzazione e caricamento dati.	30
Diagramma 3: Funzionamento generale della fase costruttiva	36
Diagramma 4: Soluzione con il minimo numero di violazioni sui ratio constraints ad alta priorità	40
Diagramma 5: Soluzione con il minimo numero di cambi di colore	48
Diagramma 6: Soluzione con il minimo numero di violazioni sui ratio constraints ad alta priorità	54
Diagramma 7: Soluzione con il minimo numero di cambi di colore	59

Strutture dati

Struttura Dati 1: Matrice	32
Struttura Dati 2: Matr_ieri	32
Struttura Dati 3: Matr_ridotta	33
Struttura Dati 4: Vettore ratios	33
Struttura Dati 5: Vettore pesi	34
Struttura Dati 6: Insieme di liste associate ad ogni posizione della sequenza	43

Figure

Figura 1: Contenuto di Help_HP	37
Figura 2: Codifiche binarie di Albero	41
Figura 3: Contenuto di matr_ridotta	41
Figura 4: Valore del campo "quanti" della struttura Albero	42
Figura 5: Contenuto di matr_help durante l'elaborazione dei dati ..	45
Figura 6: Contenuto di matr_ridotta dopo l'ordinamento	49
Figura 7: Valore dei campi del vettore tinte	50
Figura 8: Esempio di una sequenza di vetture ..	55
Figura 9: Caratteristiche ideali per la prima posizione	56
Figura 10: Vettura scelta dall'algoritmo	56

Tabelle

Tabella 1: Tecniche costruttive a confronto	61
Tabella 2: Risultati delle simulazioni	68
Tabella 3: Differenza tra i risultati ottenuti e quelli iniziali	69

Grafici

Grafico 1: Confronto degli algoritmi costruttivi sulla minimizzazione delle violazioni sui ratio constraints	62
Grafico 2: Confronto degli algoritmi costruttivi sulla minimizzazione di cambi di colore	62
Grafico 3: Confronto dei risultati ottenuti	70

Profiling

Profiling 1: Scenario 064_38_2_RAF_EP_ENP_CH1	72
Profiling 2: Scenario 022_3_4_EP_RAF_ENP	73
Profiling 3: Scenario 025_38_1_EP_ENP_RAF	74

1. Il contesto

Oggetto della tesi è lo studio di un problema di *ottimizzazione*, riguardante la schedulazione della produzione giornaliera di una fabbrica di automobili, e lo sviluppo di un software per la sua risoluzione. La schedulazione giornaliera deve tenere in considerazione alcuni vincoli che possono essere suddivisi in due differenti categorie: vincoli sulla verniciatura e vincoli sulla linea di assemblaggio.

I primi sono dovuti alla necessità di lavare le pistole spray, utilizzate per la colorazione, dopo ogni cambio di colore, o comunque regolarmente dopo un certo numero di impieghi, anche se il colore non varia. I secondi sono determinati dagli optional che le vetture possono richiedere e che impongono un maggiore carico di lavoro sulla linea di assemblaggio; questo richiede quindi di spaziare il più possibile le automobili che richiedono optional. A tal proposito gli accessori possono essere distinti tra accessori ad alta priorità, che introducono un elevato carico di lavoro, e accessori a bassa priorità, il cui peso sulla linea di assemblaggio è invece ridotto.

L'obiettivo dell'ottimizzazione consiste nel minimizzare il numero di violazioni ai vincoli, sia quelli sulla verniciatura che quelli sulla linea di assemblaggio, dando maggior peso agli uni o agli altri a seconda delle intenzioni del decisore (che danno luogo a diversi scenari per il problema). Un altro vincolo particolarmente stringente deriva dalla limitazione al tempo di esecuzione del software.

Attualmente, un'applicazione industriale si occupa del processo di pianificazione / schedulazione, usando *programmazione lineare* per associare un giorno di produzione ad ogni vettura commissionata e *simulated annealing* per determinare le sequenze di automobili da produrre ogni giorno.

Considerato che, quanto offerto dalla programmazione lineare non ha bisogno di ulteriori ottimizzazioni, lo scopo del progetto è di elaborare un metodo alternativo che fornisca soluzioni migliori, rispetto a quanto generato dalla tecnica *simulated annealing*, mantenendo inalterato l'insieme di vetture su cui operare (generato dalla programmazione lineare).

Il programma è stato prodotto nell'ambito della competizione *ROADEF CHALLENGE 2005* organizzato dalla French Society of Operations And Decision Analysis (ROADEF), il cui soggetto è stato proposto e supportato dalla Renault Automobili.

1.1 I processi di pianificazione e schedulazione

Il problema da risolvere può essere definito come *car sequencing problem*, ossia la schedulazione giornaliera della produzione di autovetture. In questo contesto gli ordini dei clienti sono trasmessi ogni giorno in tempo reale alle fabbriche di automobili.

Il lavoro giornaliero delle fabbriche è quello di assegnare un giorno di produzione ad ogni vettura ordinata, tenendo presente la capacità produttiva e le date di consegna che sono state promesse ai clienti dai venditori.

Le fabbriche devono successivamente definire l'ordine delle automobili che devono essere prodotte ogni giorno cercando di soddisfare più requisiti possibili dell'impianto: corpo vettura, verniciatura e linea di assemblaggio.

L'ottimizzazione della produzione verte sui requisiti della verniciatura e della linea di assemblaggio dato che il corpo vettura non impone requisiti per la schedulazione giornaliera. Schedulando la produzione di ogni giorno, l'insieme di vetture da produrre, non può essere cambiato.

1.1.1 Requisiti della linea di assemblaggio

Per livellare il carico di lavoro sulla linea di assemblaggio, i veicoli che richiedono speciali operazioni devono essere distribuiti attraverso l'intero insieme di auto processate. Questi veicoli sono considerati "difficili da produrre" e non possono superare una data quota su una qualsiasi sequenza.

Questi requisiti sono modellati come *Ratio Constraints N/P* (vincoli di frequenza) e sono associati alle caratteristiche delle auto che richiedono operazioni extra nella linea di assemblaggio (ad esempio aria condizionata, tetto panoramico, ecc.).

Un ratio constraint N/P richiede che al massimo N auto in ogni sequenza consecutiva di P siano associate al vincolo. Per esempio, se $N/P = 3/5$, non ci devono essere più di tre auto vincolate su ogni sequenza consecutiva di cinque veicoli.

Se $N/P = 1/P$, significa che due auto vincolate devono essere separate da almeno $P-1$ auto consecutive non vincolate. Per esempio, con un ratio constraint $1/5$: $X_ _ _ _ X$ è una sequenza accettabile, dove 'X' è un veicolo che necessita di una operazione extra in fase di assemblaggio e ' _ ' è un veicolo non riguardante il ratio constraint.

Questi vincoli di rapporto sono suddivisi in due classi: ratio constraints ad alta priorità ed a bassa priorità. Quelli ad alta priorità riguardano caratteristiche dell'auto che impongono un elevato carico di lavoro sulla linea di assemblaggio, i secondi causano solamente lievi inconvenienti alla produzione. I ratio constraints ad alta priorità devono essere soddisfatti in via preferenziale rispetto ai ratio constraints a bassa priorità.

Questi vincoli sono considerati *Soft Constraints*: il pieno rispetto di tutti i ratio constraints non può essere assicurato prima che la produzione giornaliera sia schedulata. Risulta spesso impossibile ordinare le vetture in modo che tutti i vincoli di frequenza siano soddisfatti. Per questo l'obiettivo dell'ottimizzazione è quello di minimizzare il numero di violazioni dei ratio constraints.

1.1.2 Ratio constraints ad alta priorità “facili” e “difficili” da soddisfare

Relativamente ai ratio constraints ad alta priorità, la Renault ha fornito scenari “facili” da soddisfare e scenari “difficili”. La valutazione per i due tipi di scenari è differente.

Uno scenario si considera “facile” se l'applicazione industriale attualmente utilizzata da Renault è in grado di generare una sequenza di vetture per il giorno di produzione senza nessuna violazione dei ratio constraints.

Uno scenario si considera avere ratio constraints ad alta priorità “difficile” se l'applicazione industriale Renault non riesce a generare uno schedule del giorno di produzione che rispetta tutti sopraccitati vincoli. Questo accade, ad esempio, quando il numero di veicoli, per i quali sono richieste operazioni extra in fase di assemblaggio, supera il rapporto N/P: un chiaro esempio di vincolo di

rapporto difficile da soddisfare si ottiene se il giorno di produzione contiene il 25% di veicoli vincolati e c'è un ratio constraint ad alta priorità di 1/5.

La distinzione tra scenari “facili” e “difficili” non prende in considerazione i ratio constraints di bassa priorità.

1.1.3 Requisiti della verniciatura

La verniciatura deve minimizzare il consumo di solvente. Il solvente è utilizzato per lavare le pistole spray ogni volta che il colore viene cambiato tra due vetture schedate consecutive.

Il requisito per evitare sprechi in questo settore è quello di raggruppare i veicoli per colore, così da minimizzare il numero di cambiamenti di tonalità in una sequenza di vetture schedate. In altre parole si cerca di minimizzare i lavaggi delle pistole spray, quindi di schedare sequenze di colore il più lunghe possibile.

Le sequenze di colore hanno una limitazione sul massimo numero di vetture che le compongono, dato che le pistole spray devono essere lavate regolarmente, anche se non ci sono cambiamenti di colore nella sequenza di veicoli schedati. Questa limitazione è un *Hard Constraint* (vincolo rigido), ovvero, definito l'insieme di vetture da produrre, il numero minimo di lavaggi delle pistole è determinabile e non può mai essere ridotto.

1.1.4 Ruolo del giorno di produzione precedente

Il calcolo del numero di violazioni dei ratio constraints sul giorno di produzione attuale deve tenere in considerazione gli ultimi veicoli schedulati nel giorno precedente. Questi sono già schedulati e quindi la loro posizione non può essere cambiata.

Invece, il giorno di produzione successivo, è ignorato.

1.2 Problema da risolvere

Il problema consiste nel fornire una sequenza di veicoli che soddisfi al meglio i requisiti della verniciatura e della linea di assemblaggio. A seconda della fabbrica possono essere seguite due differenti tipologie di strategie:

- Massima priorità ai ratio constraints: i requisiti della linea di assemblaggio hanno un livello di priorità più alto di quelli della verniciatura.
- Massima priorità alla verniciatura: i requisiti della verniciatura sono più critici di quelli della linea di assemblaggio.

Per ognuna delle strategie sopra descritte si possono presentare differenti scenari di ottimizzazione.

1.2.1 Ottimizzazione multi-obiettivo con massima priorità ai ratio constraints

I seguenti obiettivi devono essere ottimizzati, nell'ordine, da quello con livello di priorità più alto a quello con livello di priorità più basso, e senza alcuna compensazione tra gli obiettivi.

1. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità
2. Minimizzare il numero di cambi di colore
3. Minimizzare il numero di violazioni dei ratio constraints a bassa priorità

Oppure

1. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità
2. Minimizzare il numero di violazioni dei ratio constraints a bassa priorità
3. Minimizzare il numero di cambi di colore

Oppure

1. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità
2. Minimizzare il numero di cambi di colore

L'ordine degli obiettivi dipende dalle fabbriche, alcune delle quali non dichiarano ratio constraints di bassa priorità.

1.2.2 Ottimizzazione multi-obiettivo con massima priorità ai cambi di colore

La presente strategia prevede l'ottimizzazione dei seguenti obiettivi, dal livello di priorità più alto al più basso, e senza nessuna compensazione tra gli obiettivi, al fine di schedare i veicoli di un giorno di produzione:

1. Minimizzare il numero di cambi di colore
2. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità
3. Minimizzare il numero di violazioni dei ratio constraints a bassa priorità

Oppure

1. Minimizzare il numero di cambi di colore
2. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità

Per le due sopracitate ottimizzazioni multi-obiettivo, c'è solo un hard constraint: la limitazione alla dimensione massima di una sequenza di auto dello stesso colore; infatti le pistole spray utilizzate per la verniciatura devono essere lavate regolarmente, anche se non ci sono cambi di colore.

Non ci deve essere alcuna compensazione tra gli obiettivi: l'ottimizzazione di un obiettivo a bassa priorità non può ridurre i risultati ottenuti su quelli ad alta priorità.

1.3 Dettagli sul calcolo del numero di violazioni dei ratio constraints

Per livellare il carico di lavoro sulla linea di assemblaggio, i veicoli che richiedono speciali operazioni devono essere distribuiti attraverso l'intero insieme di auto processate.

La miglior soluzione per un ratio constraint N/P sarebbe un giorno di produzione schedato in cui ci siano al massimo N veicoli vincolati in una sequenza qualsiasi di P veicoli consecutivi. La miglior soluzione per un vincolo di rapporto $1/P$ sarebbe un giorno di produzione in cui qualsiasi coppia di auto vincolate sia separata da almeno $P - 1$ vetture non vincolate: per esempio, $X_ _ _$. X è una sequenza accettabile per un ratio constraint $1/4$, dove 'X' è un'auto con associata la limitazione in esame, mentre ' _ ' è un veicolo non considerato dal ratio constraint.

Nel caso in cui lo scenario è "facile", è necessario che le vetture che violano i vincoli di frequenza, vengano disposte in modo uniforme all'interno dell'intera sequenza di produzione del giorno in esame. Il proposito, in questo caso, è quello di cercare di limitare il più possibile il carico di lavoro nella fase di assemblaggio.

Per ottenere questa distribuzione di veicoli vincolati, è utile calcolare il totale delle violazioni, sulle limitazioni di rapporto, per ogni sequenza generata. Naturalmente, minore sarà la distanza tra le vetture che richiedono montaggi particolari, maggiori risulteranno le violazioni ed è necessario considerare che ogni vettura che richiede operazioni di montaggio speciali, ripercuoterà la sua violazione all'interno di P intervalli.

Esempio: ratio constraint $1/5 \Rightarrow$ il numero di violazioni al ratio constraint è calcolato su sequenze di 5 veicoli consecutivi

$_ _ _ X _ _ _ X$: 1 violazione nella sequenza $_ _ _ [X _ _ _ X]$
 $_ _ _ X _ _ _ X$: totale di 2 violazioni sulle sequenze consecutive $_ _ [_ X _ _ X] _$
 e $_ _ _ [X _ _ X _]$

Il numero di violazioni di un ratio constraint su una qualsiasi sequenza di P veicoli è calcolato come:

$$V = \text{Max} (NVL - N , 0)$$

$$\forall N, P$$

□

Dove

V = Numero di violazioni di un ratio constraint sulla sequenza in esame

NVL = Numero veicoli associati al ratio constraint nella sequenza

N = Numeratore del ratio constraint

P = Denominatore del ratio constraint

Esempio: ratio constraint 1/5

$_ _ X X _ _ X _$: ci sono 2 violazioni nella sequenza $[X X _ _ X]$: perché i veicoli vincolati sono NVL=3 ed il numero dei ratio constraints è N=1.

L'obiettivo dell'ottimizzazione è quello di minimizzare il numero totale di violazioni dei ratio constraints attraverso tutte le sequenze consecutive di un giorno di produzione.

Schedulando la produzione del giorno corrente, D , gli ultimi veicoli prodotti nel giorno di produzione precedente, $D - 1$, devono essere presi in considerazione per il calcolo del numero di violazioni dei ratio constraints. Questo implica che le sequenze consecutive (sulle quali il numero di violazioni dei ratio constraints è calcolato) devono iniziare il giorno di produzione $D - 1$. Per un ratio constraint N/P , la prima sequenza contiene gli ultimi $P - 1$ veicoli schedulati il giorno di produzione $D - 1$ ed il primo veicolo del giorno di produzione D . Si

deve ricordare che la sequenza di veicoli del giorno di produzione $D - 1$ è già stata fissata, quindi la loro posizione non può essere cambiata.

Esempio: la prima sequenza per un ratio constraint 1/5

XX[XXXXY] _ _ _ _

X: ultimi veicoli schedulati il giorno di produzione $D - 1$

Y: prima auto schedulata il giorno di produzione D

Dato che la produzione del giorno successivo, $D + 1$, è ignorata durante la creazione della sequenza di autovetture per il giorno D , un ratio constraint N/P , prevede che l'ultima sequenza consecutiva del giorno D contenga le ultime P auto schedulate il giorno stesso. Questa ultima sequenza consecutiva inizia nella posizione $L - P$, con L pari al numero di veicoli del giorno di produzione D ed una posizione compresa tra 0 ed $L - 1$.

Ciò significa che, per ogni ratio constraint, si devono calcolare le violazioni determinate dalle ultime auto del giorno di produzione D , come se le prime della sequenza del giorno successivo non richiedessero operazioni extra in fase di assemblaggio.

Per un ratio constraint N/P , le ultime sequenze consecutive avranno una lunghezza compresa tra $P - 1$ e $N + 1$. Se in una di queste, il numero di auto associate al ratio constraint è strettamente maggiore di N , allora ci saranno delle violazioni, sia che il primo veicolo schedulato il giorno di produzione $D + 1$ sia associato al ratio constraint sia in caso contrario.

Questo metodo di calcolo consente di dare lo stesso peso alle violazioni dovute alle ultime $P - 1$ auto del giorno di produzione D (per un ratio constraint N/P) ed a quelle dovute ai veicoli precedenti.

Esempio: le ultime sequenze consecutive per un ratio constraint 1/5

YYYY[YYYY]

YYYYYY[YYY]

YYYYYYY[YY]

[Y]: ultimi veicoli schedulati il giorno di produzione D

1.4 Dettagli sul calcolo del numero di cambi di colore

I requisiti sulla verniciatura impongono che le pistole spray siano lavate dopo ogni cambio di colore e regolarmente dopo ogni sequenza di auto consecutive dello stesso colore con una dimensione pari al limite M imposto. Il valore M è un hard constraint ed impone il lavaggio delle pistole spray ogni M veicoli con lo stesso colore.

Se la vettura successiva alla vettura M ha un colore differente viene conteggiato un solo cambio di colore, che può essere riferito indistintamente ad una delle due condizioni. Se si presenta una sequenza di X auto consecutive con lo stesso colore, con $X > M$, il numero di cambi di colore sarà dato da X / M arrotondato per difetto.

Esempio: limitazione $M = 3$

La sequenza di colori 1 1 1 2 2 2 3 3 3 3 4 4 determina 4 cambi di colore: 3 dovuti al passaggio ad un colore differente ed 1 dovuto al superamento del limite M nella sequenza 3 3 3 3.

2. Il concorso

Il software è stato prodotto nell'ambito della competizione ROADEF CHALLENGE 2005 organizzato dalla French Society of Operations And Decision Analysis (Roadeff), il cui soggetto è stato proposto e supportato dalla Renault Automobili.

La ROADEF è un'associazione nata con l'intento di promuovere la crescita della ricerca operativa e delle tecniche decisionali in Francia e di diffondere la conoscenza appresa negli ambiti industriali, fornendo i propri insegnamenti come punto di partenza per una buona formazione dei dipendenti.

Il concorso ha imposto precise regole da rispettare e ha definito, oltre ai dati su cui lavorare, i metodi di valutazione degli elaborati e la descrizione dei test set comprendente le informazioni sui formati di ingresso ed uscita. Nei capitoli successivi è descritto in modo completo quanto sopra elencato.

2.1 Linguaggi ammessi ed ambiente di test

Seguendo le specifiche fornite, il software può essere fornito come:

- un codice eseguibile che potrà essere eseguito direttamente sulla macchina fornita dalla Renault

- il codice C o C++ includendo un manuale per la compilazione, e l'associato makefile consentendo a Renault di creare un codice eseguibile.

Il tempo di esecuzione del programma è limitato a 600 secondi su un PC Pentium4/1.6Ghz/1Gb Ram. Questa limitazione rappresenta il massimo tempo di risposta accettabile da utenti industriali.

I programmi possono essere eseguiti sia in ambiente Windows2000 che RedHat Linux 7.3. Software commerciali con codici di licenza sono proibiti.

2.2 Organizzazione dei files ed interpretazione della linea di comando

Per garantire uniformità nei progetti realizzati dai diversi candidati, è stata definita un'organizzazione gerarchica dei file sulla base della seguente struttura. Per ogni partecipante, identificato come *Candidate-NN*, la gerarchia dei file è:

- Directory principale: Candidate-NN
- Sottodirectory:
 - ❖ Candidate-NN/Team-description/
 - ❖ Candidate-NN/Method-description/
 - ❖ Candidate-NN/Result-synthesis/
 - ❖ Candidate-NN/Instances/
 - ❖ Candidate-NN/Solutions/
 - ❖ Candidate-NN/Results/
 - ❖ Candidate-NN/Program/

L'eseguibile fornito, inoltre, deve essere utilizzabile digitando nella directory Candidate-NN/Program/ :

nome-dell'eseguibile nome_dell'istanza -t tempo_di_CPU

con:

nome_dell'istanza: nome dello scenario

tempo_di_CPU: tempo di esecuzione allocato in secondi, per default
fissato a 600 secondi

Il programma trova i dati di input dello scenario '*nome_dell'istanza*' nella directory Candidate-NN/Instances/nome_dell'istanza e scrive la soluzione ottenuta dopo '*tempo_di_CPU*' secondi nella directory Candidate-NN/Solutions/. Il file soluzione deve avere nome '*nome_dell'istanza*'.

Nel caso in cui non venga specificato il tempo di esecuzione, si assume come tempo di default 600 secondi.

2.3 Procedure di valutazione e classificazione

Tutti i partecipanti sono suddivisi in due categorie per la valutazione finale:

- Senior: tutti i candidati appartengono a questa categoria
- Junior: limitato a progetti di studenti (un progetto è definito "di studenti" se la maggior parte dei membri della squadra sono studenti, possibilmente seguiti dai propri professori)

Nota:

Da notare che un progetto di studenti può vincere il premio della categoria senior se il progetto è il migliore tra tutti i progetti inviati. Al contrario un progetto senior non può vincere il premio della categoria junior.

2.3.1 Il programma di valutazione

Il programma di valutazione Renault, per ogni scenario, richiama il programma del candidato, lo lascia eseguire per 600 secondi, cerca il file soluzione ottenuto nella directory *Candidate-NN/Solutions/*, lo controlla e lo valuta e scrive la valutazione della soluzione nella directory *Candidate-NN/Results/*.

Per programmi non deterministici, sono richieste 10 esecuzioni per scenario per ottenere la soluzione media.

2.3.2 Metodo di valutazione di uno scenario

Il metodo di valutazione per le qualificazioni è indicato di seguito. Nel metodo di valutazione per le finali potranno essere apportati piccoli cambiamenti. Per ogni scenario viene calcolato un punteggio, pesando i differenti obiettivi di ottimizzazione dello scenario.

Scenario con massima priorità ai ratio constraints:

Sequenza degli obiettivi di ottimizzazione:

1. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità
2. Minimizzare il numero di cambi di colore

3. Minimizzare il numero di violazioni dei ratio constraints a bassa priorità

Punteggio dello scenario

$$\begin{aligned} &= (10000 * \text{numero di violazioni dei ratio constraints ad alta priorità}) \\ &+ (100 * \text{numero di cambi di colore}) \\ &+ (\text{numero di violazioni dei ratio constraints a bassa priorità}) \end{aligned}$$

Oppure

1. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità
2. Minimizzare il numero di violazioni dei ratio constraints a bassa priorità
3. Minimizzare il numero di cambi di colore

Punteggio dello scenario

$$\begin{aligned} &= (10000 * \text{numero di violazioni dei ratio constraints ad alta priorità}) \\ &+ (100 * \text{numero di violazioni dei ratio constraints a bassa priorità}) \\ &+ (\text{numero di cambi di colore}) \end{aligned}$$

Oppure

1. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità
2. Minimizzare il numero di cambi di colore

Punteggio dello scenario

$$\begin{aligned} &= (10000 * \text{numero di violazioni dei ratio constraints ad alta priorità}) \\ &+ (100 * \text{numero di cambi di colore}) \end{aligned}$$

Scenario con massima priorità ai cambi di colore:

Sequenza degli obiettivi di ottimizzazione:

1. Minimizzare il numero di cambi di colore
2. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità
3. Minimizzare il numero di violazioni dei ratio constraints a bassa priorità

Punteggio dello scenario

$$\begin{aligned} &= (10000 * \text{numero di cambi di colore}) \\ &+ (100 * \text{numero di violazioni dei ratio constraints ad alta priorità}) \\ &+ (\text{numero di violazioni dei ratio constraints a bassa priorità}) \end{aligned}$$

Oppure

1. Minimizzare il numero di cambi di colore
2. Minimizzare il numero di violazioni dei ratio constraints ad alta priorità

Punteggio dello scenario

$$\begin{aligned} &= (10000 * \text{numero di cambi di colore}) \\ &+ (100 * \text{numero di violazioni dei ratio constraints ad alta priorità}) \end{aligned}$$

2.3.3 Valutazione globale di un test set

Per ogni test set ci sono 3 classi di scenari:

- scenari con massima priorità ai ratio constraints e ratio constraints ad alta priorità facili da soddisfare
- scenari con massima priorità ai ratio constraints e ratio constraints ad alta priorità difficili da soddisfare
- scenari con massima priorità ai cambi di colore

Per ogni classe di scenari è calcolata una media dei punteggi ottenuti su tutti gli scenari appartenenti alla stessa classe (il punteggio di ogni scenario è calcolato utilizzando il metodo discusso in precedenza).

Attraverso le tre classi, viene calcolato un punteggio globale per valutare il candidato:

$$\sum_i \frac{C_i - P_i}{M_i - P_i}$$

□

Dove:

C_i = punteggio del candidato sulla classe i

P_i = punteggio peggiore della classe i

M_i = punteggio migliore della classe i

i = classe degli scenari

Il punteggio peggiore ed il punteggio migliore sono il peggiore di tutti i punteggi ottenuti dal candidato, e il migliore di tutti i punteggi ottenuti dal candidato, per una data classe di scenari.

Il punteggio globale è l'unico criterio per classificare i candidati. Questo metodo di valutazione dovrebbe eliminare qualsiasi gap numerico tra i punteggi medi delle tre classi di scenari.

Basandosi sui risultati del test set A, un massimo di 10 candidati saranno selezionati per le finali. I finalisti saranno classificati sulla base dei risultati sul test set X. Il test set B è fornito solamente per mettere a punto i programmi dei candidati.

2.4 Descrizione dei test set

Un test set è rappresentato da un insieme di files che descrivono differenti scenari. Il software sviluppato deve caricare i files corrispondenti allo scenario specificato e procedere all'opportuna ottimizzazione in base alle caratteristiche specifiche dello scenario stesso. La Renault ha fornito 3 differenti test set, da utilizzare per lo sviluppo e l'ottimizzazione del software.

2.4.1 Contenuto di un test set

Un test set contiene un insieme di scenari, ognuno dei quali rappresenta i dati di produzione per una fabbrica. La stessa fabbrica può essere utilizzata per produrre diversi scenari.

Definito D il giorno di produzione attuale, uno scenario contiene:

- Ratio constraints ad alta priorità
- Ratio constraints a bassa priorità
- Limite al numero di auto consecutive con lo stesso colore
- Veicoli del giorno di produzione D e le ultime auto schedate del giorno di produzione $D - 1$ (ci sono $\text{DenMax} - 1$ veicoli del giorno di produzione $D - 1$, con DenMax il massimo dei denominatori dei ratio constraints)
- Un veicolo è identificato da: un ID, la data del giorno di produzione (data D o data $D - 1$), il colore della verniciatura ed un flag (0/1) per ogni ratio constraint che indica se il veicolo è associato o meno al vincolo, ed il numero di sequenza schedato dall'applicazione industriale Renault.
- L'ordine degli obiettivi di ottimizzazione.

Alcuni scenari possono non contenere nessun ratio constraint di bassa priorità in quanto alcune fabbriche possono dare alta priorità a tutti oppure non avere ratio constraints a bassa priorità.

2.5 Formato dei files

Uno scenario è una directory chiamata `instance_XXXXXX` contenente i seguenti files:

- file della limitazione delle sequenze di auto con lo stesso colore <<`paint_batch_limit.txt`>>
- file degli obiettivi di ottimizzazione <<`optimization_objectives.txt`>>
- file dei ratio constraints <<`ratios.txt`>>
- file dei veicoli <<`vehicles.txt`>>

Essi sono tutti files di testo ASCII che hanno il carattere ‘;’ come delimitatore.

2.5.1 File della limitazione delle sequenze di auto con lo stesso colore

Questo file contiene il limite alla dimensione di sequenze di auto consecutive con lo stesso colore, ossia ogni quante auto consecutive dello stesso colore si deve procedere al lavaggio delle pistole spray con il solvente.

Esempio del contenuto di un file <<`paint_batch_limit.txt`>>

limitation;

10;

2.5.2 File degli obiettivi di ottimizzazione

Il file contiene l'ordine degli obiettivi di ottimizzazione, dall'obiettivo con il più alto livello di priorità a quello con il livello di priorità più basso. Il file contiene una lista di valori del tipo *<<posizione; nome dell'obiettivo>>*. Il nome dell'obiettivo indica il tipo (cambi di colore o ratio constraint) e se i ratio constraints ad alta priorità sono facili da soddisfare o difficili da soddisfare.

Diversi esempi del contenuto del file *<<optimization_objectives.txt>>*:

Esempio del contenuto del file nel caso di massima priorità ai ratio constraints e ratio constraints ad alta priorità facili da soddisfare:

```
rank;objective name;  
1;high_priority_level_and_easy_to_satisfy_ratio_constraints;  
2;paint_color_batches;  
3;low_priority_level_ratio_constraints;
```

Esempio del contenuto del file in caso di massima priorità ai cambi di colore e ratio constraints ad alta priorità facili da soddisfare.

```
rank;objective name;  
1;paint_color_batches;  
2;high_priority_level_and_easy_to_satisfy_ratio_constraints;
```

Possibile contenuto del file in caso di massima priorità ai ratio constraints e ratio constraints ad alta priorità difficili da soddisfare

```
rank;objective name;  
1;high_priority_level_and_difficult_to_satisfy_ratio_constraints;  
2;paint_color_batches;
```

3;low_priority_level_ratio_constraints;

2.5.3 File dei ratio constraints

Il file descrive ratio constraints di alta priorità e bassa priorità. Ognuno di essi è definito da un rapporto N/P (N numero massimo di vetture associato al vincolo di rapporto su una qualsiasi sequenza di P veicoli), un flag per il livello di priorità (1=alta priorità, 0=bassa priorità) ed un identificatore (la posizione del ratio constraint).

Esempio del contenuto del file <<ratios.txt>>

```
Ratio;Prio; Ident;
1/2; 1;   HPRC1;
1/8; 1;   HPRC2;
2/3; 1;   HPRC3;
1/4; 1;   HPRC4;
3/4; 0;   LPRC1;
1/3; 0;   LPRC2;
2/3; 0;   LPRC3;
```

In questo esempio ci sono quattro ratio constraints ad alta priorità e tre a bassa priorità.

2.5.4 File dei veicoli

Il file contiene i veicoli della produzione odierna (D) e le ultime auto schedate il giorno di produzione precedente ($D - 1$).

Per ogni veicolo sono riportati i seguenti campi:

1. *Data*: il giorno di produzione del veicolo, col formato “AAAA WW D”, dove AAAA è l’anno, WW è la settimana industriale nell’anno (WW = 1..52), e D è il giorno della settimana (1 per lunedì, 2 per martedì, ecc.)
2. *Numero di sequenza*: la posizione del veicolo nel giorno di produzione. Questa posizione è definita dall’applicazione industriale Renault. Le posizioni in un giorno di produzione iniziano dal numero 1.
3. *ID*: identificativo del veicolo.
4. *Colore*: codice del colore della verniciatura del veicolo.
5. *HPRCi*: ‘1’ se il veicolo è associato al ratio constraint ad alta priorità HPRCi, ‘0’ altrimenti. Il numero di colonne HPRCi è uguale al numero di ratio constraints ad alta priorità.
6. *LPRCi*: la stessa situazione, ma con ratio constraints a bassa priorità.

Solo gli ultimi veicoli schedulati il giorno di produzione $D - 1$ sono forniti in quanto solo queste ultime auto sono necessarie per calcolare il numero di violazioni dei ratio constraints nel giorno D . Il numero di veicoli del giorno precedente è uguale al massimo dei denominatori dei ratio constraints $- 1$.

Esempio del contenuto del file <<*vehicles.txt*>>

Date;SeqRank;Ident;Paint Color;HPRC1; HPRC2;

HPRC3;HPRC4;LPRC1;LPRC2; LPRC3

```
2003 18 3;1001;025031850094;2;0;0;0;0;0;0
2003 18 3;1002;025031910010;2;1;1;0;0;1;0
2003 18 3;1003;025031850332;2;0;0;0;0;1;0
2003 18 3;1004;025031820317;1;1;0;1;1;1;1
2003 18 5;1;025031910825;3;0;0;1;0;1;0;1;0
2003 18 5;2;025031820293;7;1;0;1;1;1;0;1;0
2003 18 5;3;025031850497;2;0;0;0;0;0;0;0;0
2003 18 5;4;025031911085;5;1;0;1;0;1;1;1;0
2003 18 5;5;025031850299;5;0;0;1;0;1;0;1;0
2003 18 5;6;025031850082;2;1;0;0;0;0;0;0;0
```

```

2003 18 5;7;025031850783;5;0;0;1;0;1;1;1;0
2003 18 5;8;025031920415;5;1;1;1;1;0;0;1;0
2003 18 5;9;025032010270;5;0;0;0;0;0;0;0;0
2003 18 5;10;025031911083;5;1;0;1;0;1;1;1
2003 18 5;11;025031910060;5;0;0;0;0;0;0;0;1
2003 18 5;12;025031970126;5;1;0;1;1;0;0;1
2003 18 5;13;025031910182;2;0;0;0;0;1;0;0

```

Nell'esempio sopra riportato, ci sono 4 ratio constraint ad alta priorità, 3 ratio constraints a bassa priorità, 4 veicoli del giorno di produzione D – 1 e 13 auto del giorno di produzione D.

Data del giorno di produzione D – 1 <<2003 18 3>>

Data del giorno di produzione D <<2003 18 5>>

Come dimostrato nell'esempio, le date dei giorni di produzione D e D – 1 non sono sempre consecutive. Esse riguardano esclusivamente i giorni lavorativi nelle fabbriche, senza domeniche e festività. Nell'esempio la data <<2003 18 4>> è il 1° Maggio 2003 che è una festività.

Per ogni veicolo, i campi dei ratio constraints ad alta priorità sono definiti prima dei campi dei ratio constraints a bassa priorità. La sequenza dei campi appartenenti alle limitazioni di rapporto è identica alla sequenza dei ratio constraints nel file <<ratios.txt>>.

I veicoli del giorno di produzione D – 1 sono definiti prima delle auto del giorno di produzione D.

2.5.5 File delle soluzioni

I files delle soluzioni, contengono le sequenza di vetture generate ed ottimizzate al meglio dal software, corrispondenti ai differenti scenari che devono essere fornite.

Un file soluzione deve avere i seguenti campi: <<*Numero di sequenza; ID*>> con il numero di sequenza che inizia da 1. Il file soluzione dovrà avere lo stesso nome dello scenario.

Esempio del contenuto di un file soluzione

```
1;025031910825  
2;025031820293  
3;025031850497  
4;025031911085  
5;025031850299  
6;025031850082  
7;025031850783  
8;025031920415  
9;025032010270  
10;025031911083  
11;025031910060  
12;025031970126
```

Il file di output riportato nell'esempio, contiene la sequenza schedulata dal software per il giorno di produzione D contenente 12 vetture, per ognuna delle quali viene inserito il numero di sequenza.

3. Struttura generale

La scelta delle tecniche risolutive è stata fortemente influenzata dalla limitazione imposta sul tempo di esecuzione del software. Infatti, come descritto precedentemente, il programma sarà eseguito su una macchina di riferimento (P4 1.6Ghz / 1Gb Ram) per un periodo di tempo pari a 600 secondi. Per far fronte a questo vincolo sono stati implementati algoritmi che consentono un'esecuzione rapida dell'ottimizzazione ed anche le strutture dati di supporto sono state definite in modo tale da permetterne un utilizzo ed un aggiornamento rapido.

Anche la scelta del linguaggio in cui il software è stato implementato, è stata vincolata dalla limitazione di tempo imposta.

Il software è infatti stato implementato utilizzando il linguaggio C che, se da un lato complica la scrittura del codice, dall'altro offre un'elevata velocità di esecuzione essendo un linguaggio particolarmente efficiente, quasi a basso livello. Come ambiente di sviluppo, essendo vietati prodotti commerciali, è stato utilizzato Dev-C++ prodotto dalla software house Bloodshed. Questo software si basa sul compilatore gratuito GCC 3.2 che è stato utilizzato sfruttandone le opzioni di ottimizzazione del codice.

3.1 Diagramma di flusso globale

Il funzionamento generale del software può essere rappresentato mediante un diagramma di flusso che riportiamo di seguito:

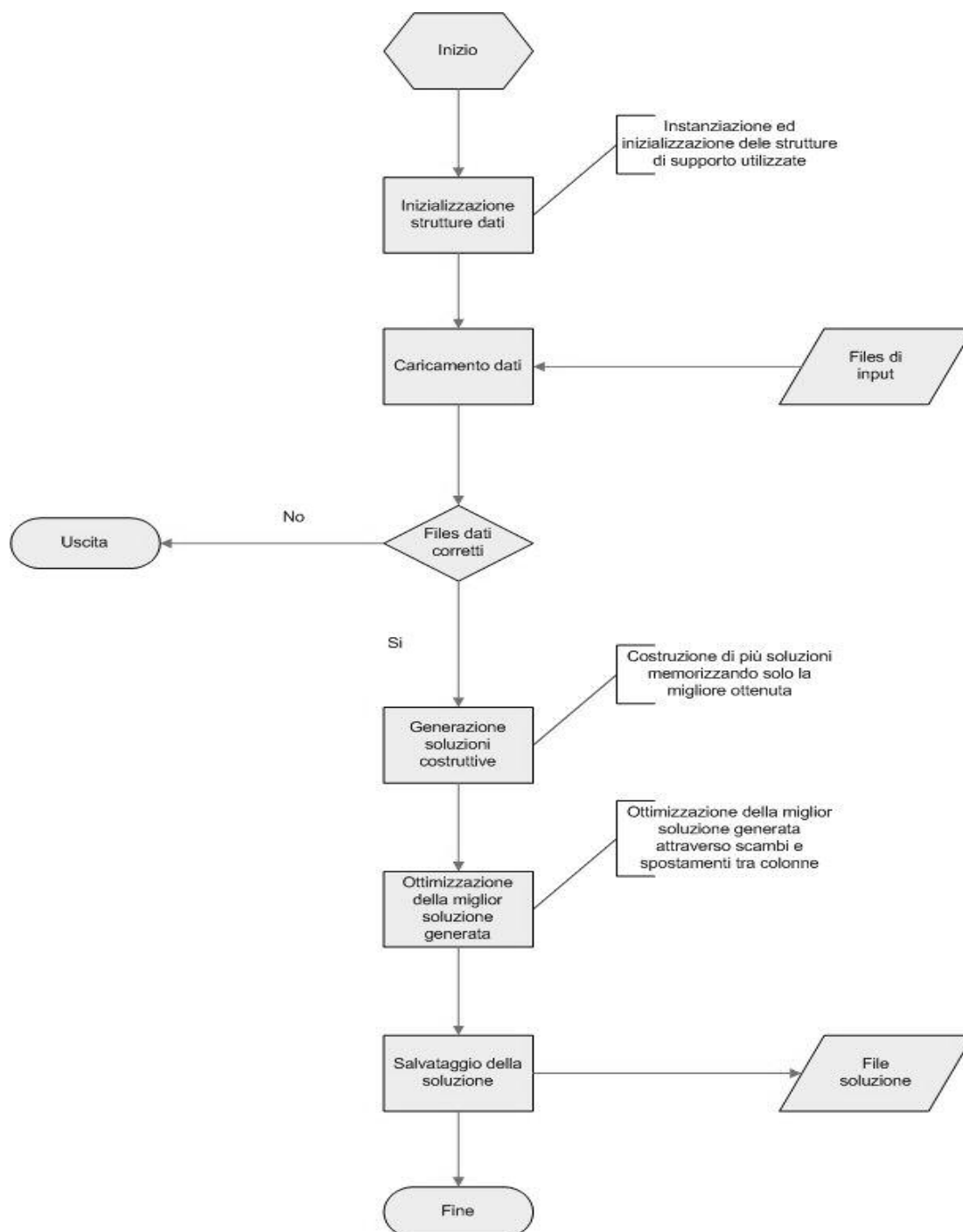


Diagramma 1: Funzionamento generale

Analizzando il diagramma è possibile notare la suddivisione del software in quattro parti fondamentali:

- Inizializzazione e caricamento dati
- Generazione di molteplici soluzioni con tecnica costruttiva
- Ottimizzazione della miglior soluzione generata mediante ricerca locale
- Salvataggio della soluzione

3.2 Inizializzazione e caricamento dati

La fase iniziale è rappresentata dalla definizione ed inizializzazione delle strutture utilizzate, nonché dal caricamento dei dati. Il funzionamento del software in questo primo approccio può essere rappresentato come segue:

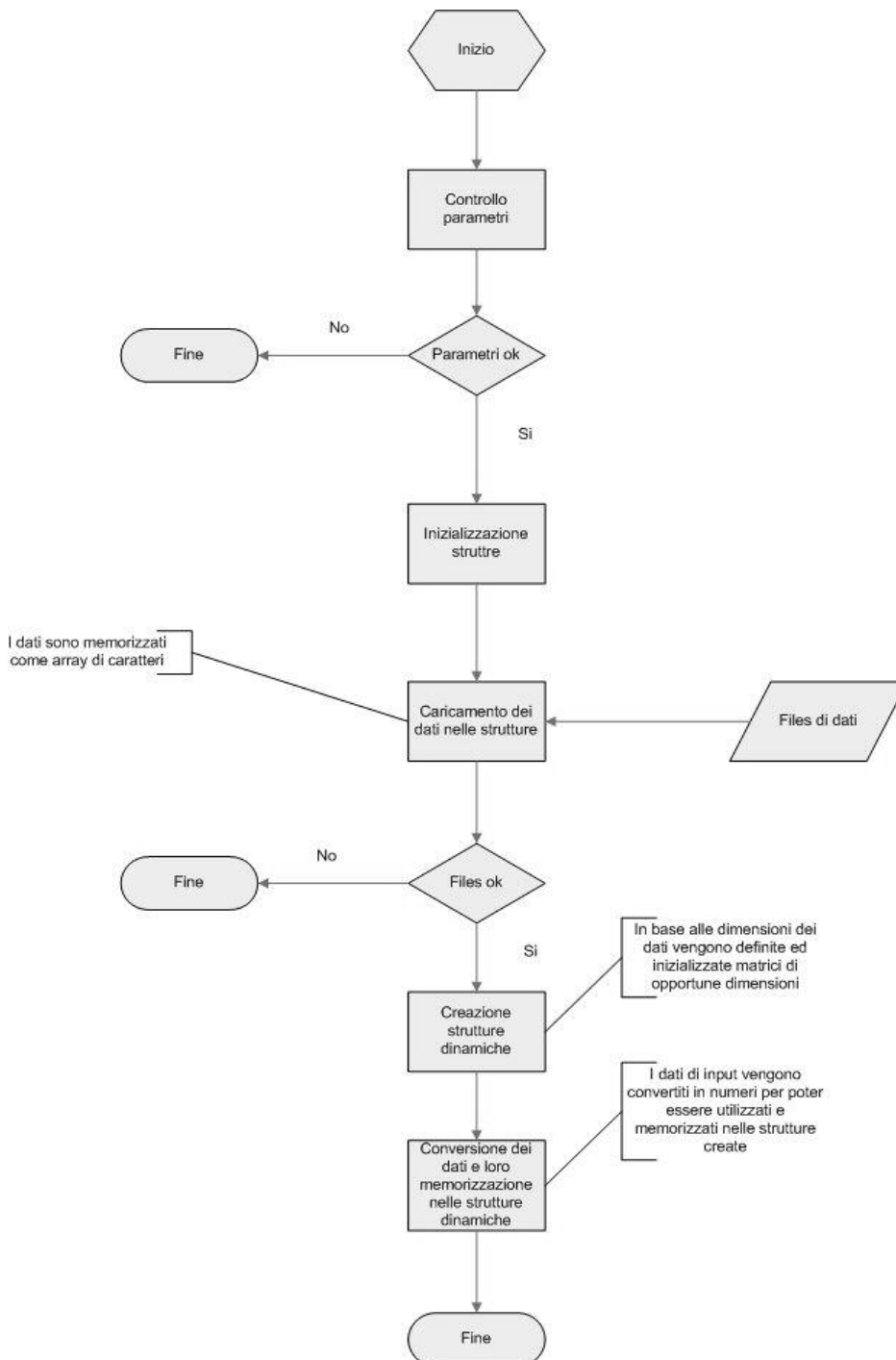


Diagramma 2: Inizializzazione e caricamento dati

Il primo passo effettuato consiste nel controllo dei parametri di input, in quanto deve essere possibile specificare il nome dello scenario i cui dati vengono processati nonché il tempo massimo di elaborazione ammesso. Quest'ultimo parametro è opzionale, in quanto se omissivo viene utilizzato il valore di default del timeout pari a 600 secondi (10 minuti). L'inserimento di parametri errati determina l'immediata uscita dal software senza eseguire alcuna elaborazione.

Se i parametri inseriti sono corretti il software procede alla fase successiva rappresentata dalla lettura dei files di dati relativi allo scenario specificato. Come riportato in precedenza i dati sono contenuti in quattro files di testo ASCII, ognuno dei quali viene letto e memorizzato in opportune matrici costituite da array di caratteri. Anche in questa fase la presenza di files il cui percorso, o nome, inserito non risulta corretto determina l'interruzione del programma e la comunicazione del file di input errato.

Sulla base delle dimensioni dei files vengono create dinamicamente le strutture dati necessarie all'elaborazione rappresentate da array e matrici di numeri interi. Dopo la creazione, ogni elemento di queste strutture viene inizializzato con un valore convenzionale che non sarà mai presente nei dati forniti. I dati letti in precedenza e memorizzati come caratteri sono quindi convertiti in numeri interi e salvati nelle strutture dinamiche appena create.

3.2.1 Strutture dati

I dati di input sono rappresentati da quattro files di testo ASCII ognuno dei quali contiene informazioni che sono memorizzate in una differente struttura dati.

In base al file di input si possono distinguere le strutture necessarie che sono descritte di seguito:

- <<vehicles.txt>>: fornisce le informazioni relative alle vetture che vengono memorizzate nelle seguenti matrici:

❖ *Matrice*: struttura rappresentata da una matrice in cui ogni posizione è a sua volta un array di caratteri. I dati letti inizialmente sono salvati in questa matrice prima di essere convertiti in numeri e memorizzati in strutture idonee.

<i>Auto 1</i>	<i>Auto 2</i>	<i>Auto 3</i>	<i>Auto 4</i>
Day	Day	Day	Day
ID	ID	ID	ID
N°	N°	N°	N°
Colore	Colore	Colore	Colore
Flag Opt. 1	Flag Opt. 1	Flag Opt. 1	Flag Opt. 1
Flag Opt. 2	Flag Opt. 2	Flag Opt. 2	Flag Opt. 2
Flag Opt. 3	Flag Opt. 3	Flag Opt. 3	Flag Opt. 3
.....

Struttura Dati 1: Matrice

❖ *Matr_Ieri*: per ogni veicolo appartenente al giorno di produzione D – 1 sono salvati ID, Colore e valori relativi agli optional.

<i>Auto 1</i>	<i>Auto 2</i>	<i>Auto 3</i>	<i>Auto 4</i>
ID	ID	ID	ID
Colore	Colore	Colore	Colore
Flag Opt. 1	Flag Opt. 1	Flag Opt. 1	Flag Opt. 1
Flag Opt. 2	Flag Opt. 2	Flag Opt. 2	Flag Opt. 2
Flag Opt. 3	Flag Opt. 3	Flag Opt. 3	Flag Opt. 3
.....

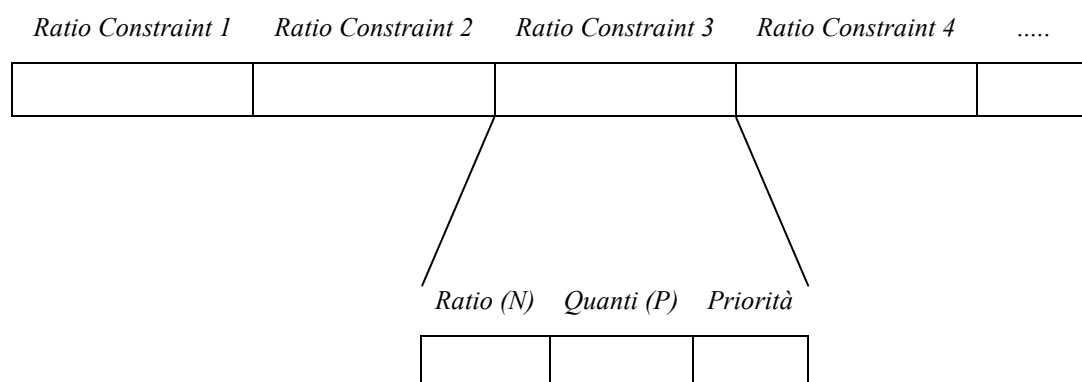
Struttura Dati 2: Matr_ieri

- ❖ *Matr_ridotta*: stesse informazioni di *matr_ieri* ma riferite ai veicoli del giorno di produzione D.

<i>Auto 1</i>	<i>Auto 2</i>	<i>Auto 3</i>	<i>Auto 4</i>
ID	ID	ID	ID
Colore	Colore	Colore	Colore
Flag Opt. 1	Flag Opt. 1	Flag Opt. 1	Flag Opt. 1
Flag Opt. 2	Flag Opt. 2	Flag Opt. 2	Flag Opt. 2
Flag Opt. 3	Flag Opt. 3	Flag Opt. 3	Flag Opt. 3
.....

Struttura Dati 3: Matr_ridotta

- <<paint_batch_limit.txt>>: contiene la limitazione alla dimensione massima di una sequenza di auto consecutive senza che questa richieda il lavaggio delle pistole spray. Il valore fornito è convertito in numero intero e memorizzato in una semplice variabile intera chiamata *limit_brush*.
- <<ratios.txt>>: il file specifica i ratio constraints di tipo N/P per ognuno dei quali sono indicati i valori di N e P nonché la priorità. Queste informazioni sono memorizzate in un array chiamato *ratios* in cui ogni posizione è composta da una struttura dati contenente:
 - ❖ *Ratio*: campo intero corrispondente al valore P
 - ❖ *Quanti*: il valore N, anch'esso di tipo intero
 - ❖ *Prio*: indica la priorità del ratio constraint. Un valore pari a '1' rappresenta un ratio constraint ad alta priorità mentre lo '0' coincide con ratio constrain a bassa priorità.



Struttura Dati 4: Vettore ratios

- <<optimization_objectives.txt>>: specifica quali sono gli obiettivi di ottimizzazione nonché l'ordine in cui questi obiettivi devono essere raggiunti. Ciò è memorizzato in un vettore chiamato *pesi* formato da 3 numeri interi che indicano il peso di ciascun obiettivo secondo il metodo di valutazione riportato in seguito. In particolare il primo obiettivo avrà peso 10000, il secondo 100 mentre il terzo, se presente, 1. La corrispondenza tra tipo di obiettivo e posizione nell'array *pesi* è la seguente:

- ❖ *Pesi*[0] → cambi di colore
- ❖ *Pesi*[1] → ratio constraint ad alta priorità
- ❖ *Pesi*[2] → ratio constraint a bassa priorità

Pesi

<i>Posizione</i>	<i>Contenuto</i>
[0]	Peso dell'obiettivo cambi di colore
[1]	Peso dei ratio constraints ad alta priorità
[2]	Peso dei ratio constraints a bassa priorità

Struttura Dati 5: Vettore *pesi*

4. La fase costruttiva

Il problema viene inizialmente affrontato con un algoritmo di tipo costruttivo. La caratteristica principale di questa tipologia di algoritmi è che a partire da una soluzione vuota, vi aggiungono iterativamente elementi fino ad ottenere una soluzione completa. Nel nostro caso, partendo da una soluzione iniziale senza alcuna vettura utilizzata, si genera una sequenza di automobili, che rispetti il più possibile i vincoli imposti.

L'algoritmo implementato attraverso questa tecnica costruttiva, è di tipo *greedy* (vorace), ovvero determina, ad ogni passo, l'elemento da aggiungere attraverso una sequenza di decisioni "localmente ottime", senza mai ritornare, modificandole, sulle decisioni prese.

Un algoritmo di questa tipologia ordina dapprima tutti gli oggetti da valutare secondo un criterio di appetibilità e costruisce la soluzione in modo incrementale, considerando gli oggetti uno alla volta, ed aggiungendo, se possibile, l'oggetto più appetibile, cioè quello che fornisce immediatamente il miglior risultato.

Fatta la scelta, l'algoritmo, si disporrà a risolvere un sottoproblema dello stesso tipo, ma naturalmente di dimensione inferiore, le cui decisioni dipenderanno esclusivamente dalle scelte fatte in precedenza, mai da quelle future. Ovviamente, un algoritmo greedy non è in grado di trovare la soluzione ottima per qualsiasi problema di ottimizzazione.

Il diagramma 3 mostra il flusso generale delle operazioni eseguite nella fase costruttiva.

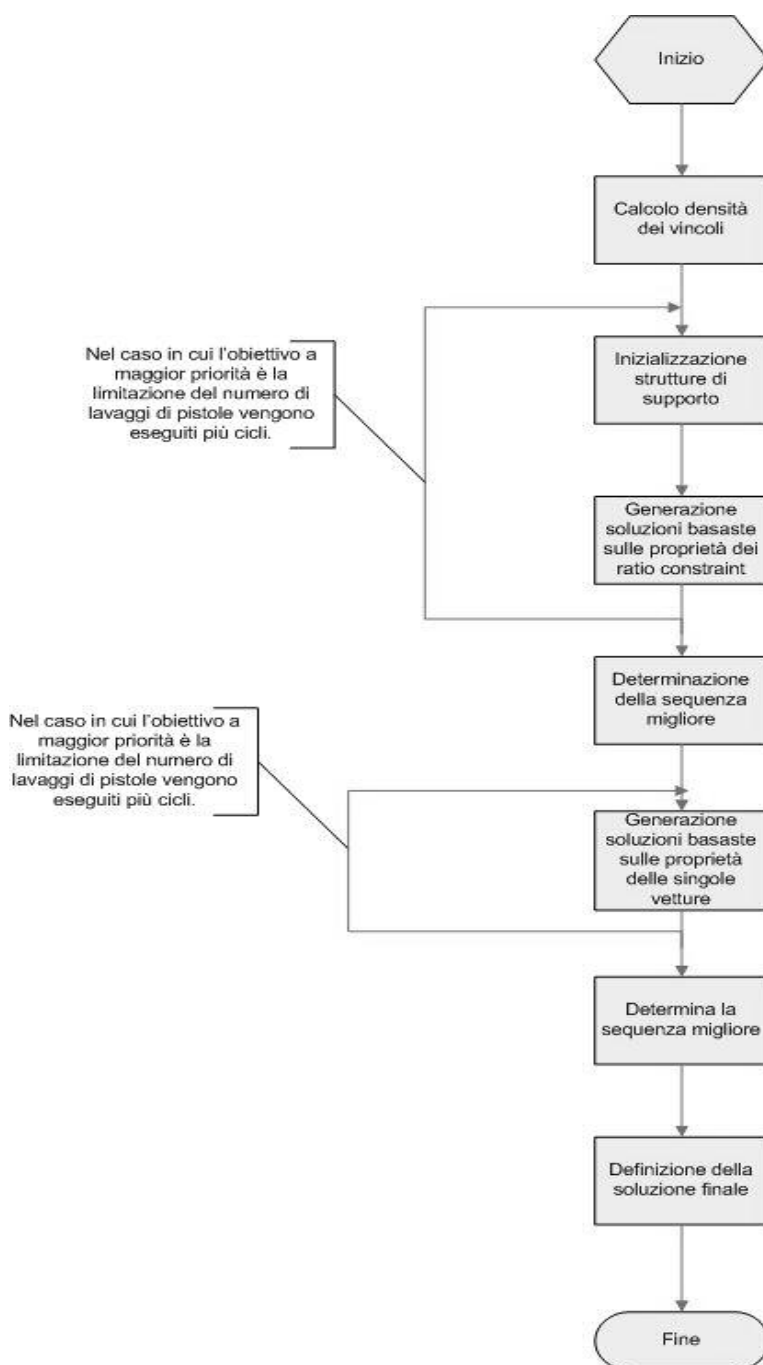


Diagramma 3: Funzionamento generale della fase costruttiva

4.1 Generazione della sequenza finale

Per determinare una sequenza di vetture ammissibile, l'algoritmo si preoccupa inizialmente di ordinare i vincoli imposti nella fase di assemblaggio in base alla densità delle diverse richieste.

Al termine di questa fase, l'algoritmo si suddivide in due grandi sezioni:

- Generazione della soluzione per righe
- Generazione della soluzione per colonne

Nessuna delle due sezioni richiede obbligatoriamente la presenza dell'altra, ma abbiamo deciso di eseguirle entrambe ogni volta per avere più possibilità di produrre la soluzione migliore.

La sequenza di vetture finale, generata dalla fase costruttiva, non sarà quella definitiva, in quanto, ciò che viene generato, sarà ulteriormente ottimizzato da un algoritmo di ricerca locale descritto nei capitoli successivi.

4.2 Inizializzazione delle strutture e calcolo della densità dei vincoli

In questa prima fase l'algoritmo conta, per ogni ratio constraint, il numero di vetture che richiedono quella particolare operazione nella fase di assemblaggio e si determina la densità delle richieste di tale optional utilizzando la formula seguente:

$$M \cdot P / H \cdot N$$

□

Dove M è il numero di vetture che impongono alla fase di assemblaggio il ratio constraint in esame e P è la larghezza massima della sequenza in cui conteggiare le violazioni sul vincolo. Il prodotto di queste due grandezze, rapportato a quello ottenuto tra il totale H di vetture da produrre il giorno D ed il numero di veicoli vincolati in una sequenza qualsiasi di P auto consecutive (sopra definito come N), restituisce la densità di vetture che causeranno un lavoro straordinario di quel tipo.

Per tutti i vincoli in cui questo valore risulta superiore ad 1 è certo che il totale delle violazioni nella sequenza finale sarà maggiore di zero. Un ratio constraint con densità superiore ad 1, infatti, rivela che il numero di vetture che lo richiedono rapportato al totale delle automobili è superiore al rapporto N/P .

Per ogni ratio constraint viene infine stilata una graduatoria in base al valore della densità. Questa graduatoria è contenuta, nelle strutture *Help_HP* ed *Help_LP*.

Un esempio del contenuto della prima è riportato di seguito.

<i>Help_HP</i>	<table><tr><td><i>1</i></td><td><i>0</i></td><td><i>2</i></td><td><i>3</i></td></tr></table>	<i>1</i>	<i>0</i>	<i>2</i>	<i>3</i>
<i>1</i>	<i>0</i>	<i>2</i>	<i>3</i>		

Figura 1: Esempio del contenuto di *Help_HP*

La figura riportata rivela che il ratio constraint ad alta priorità con densità più vicina ad uno, in uno scenario con quattro vincoli di questa categoria, è quello con indice 1, seguito dal vincolo con indice 0 e così via.

4.3 Generazione della soluzione per righe

Questa soluzione viene generata determinando, per ogni posizione della sequenza, un insieme di vetture ammissibili, stabilendo quali valori dei vincoli di

rapporto, e dei colori, è opportuno che le vetture possiedano, per poter essere collocate in tale locazione in modo da ottimizzare gli obiettivi. Naturalmente l'algoritmo richiederà elaborazioni differenti a seconda che l'obiettivo a maggior priorità sia la minimizzazione delle violazioni sui ratio constraints ad alta priorità oppure quello del minimo numero di cambi di colore.

4.3.1 Generazione della soluzione basata sul minimo numero di violazioni sui ratio constraints ad alta priorità

Gli scenari in cui l'obiettivo fondamentale è minimizzare il numero di violazioni sui vincoli di rapporto ad alta priorità, hanno come caratteristica fondamentale che, inizialmente, l'algoritmo potrebbe porre ogni vettura in qualsiasi posizione della sequenza.

Il diagramma 4 descrive le operazioni eseguite in questa sezione dell'algoritmo.

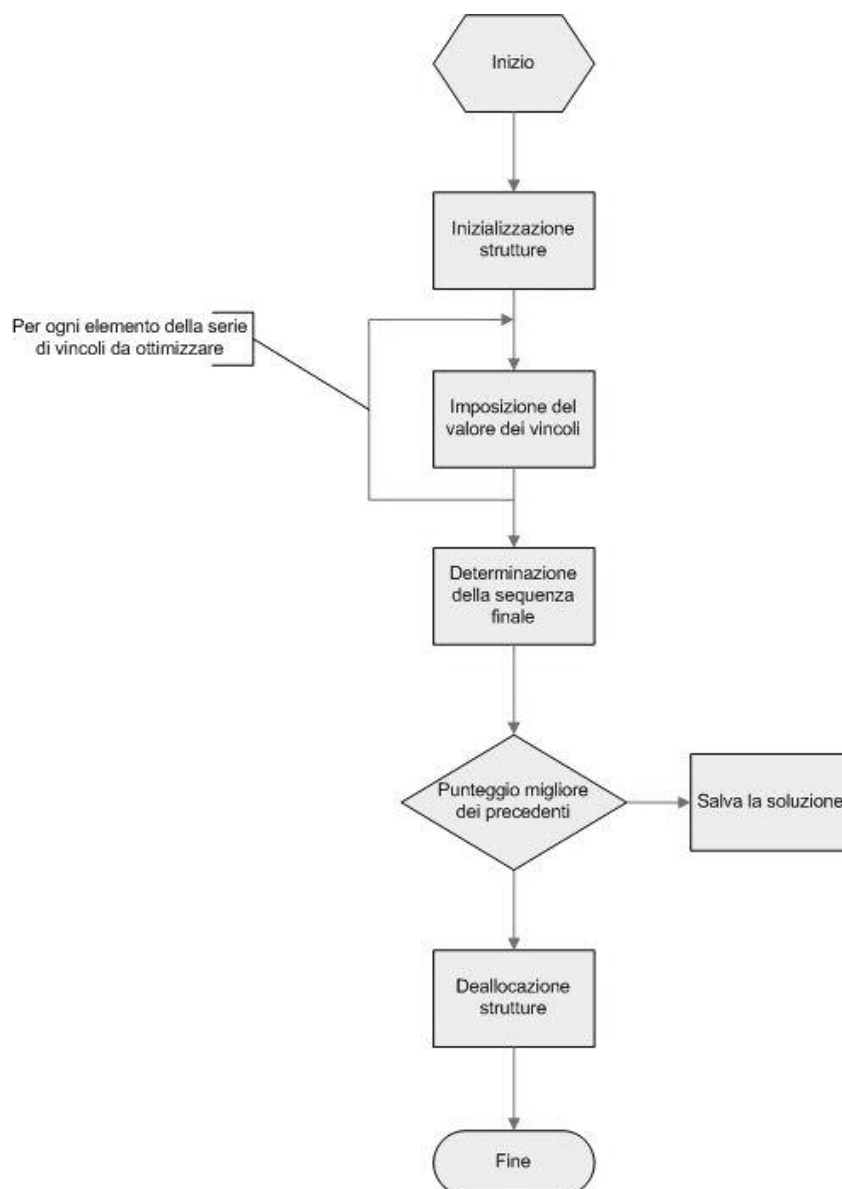


Diagramma 4: Soluzione con il minimo numero di violazioni sui ratio constraints ad alta priorità

Anzitutto si genera la struttura *Albero* che è costituita da un numero di righe pari al numero di vincoli di rapporto considerati. Ogni riga ha un indice numerico, da 0 a N , ed è composta da un numero di elementi pari a 2^{K+1} dove K è l'indice di riga.

L'origine di questa struttura dati deriva dal fatto che i ratio constraints imposti da ogni vettura, essendo composti esclusivamente da zeri ed uno, possono essere interpretati come la codifica binaria di una posizione all'interno di *Albero*.

La riga a cui riferirsi è dettata dall'ordine cronologico di scansione dei vincoli, l'elemento della riga effettuando la conversione da binario a decimale dei valori dei ratio constraints fissati in precedenza, compreso quello della riga in esame.

Di seguito è riportato un esempio dove per ogni elemento delle righe, in questo caso tre, sono riportate le codifiche binarie che permettono di accedere a tale unità della riga in esame.

0	1						
00	01	10	11				
000	001	010	011	100	101	110	111

Figura 2: Codifiche binarie delle diverse posizioni di *Albero*

Per ogni elemento di questa struttura, vengono memorizzati due attributi, uno che descrive quante vetture possiedono quella codifica binaria dei vincoli interessati ed un secondo che descrive quante di queste vetture hanno già una posizione fissa nella sequenza.

Il primo attributo di ogni elemento di *Albero*, si inizializza in base al numero di vetture che possiedono la codifica corrispondente. Un esempio può rendere senz'altro più comprensibile questa fase. Supponendo di avere la sequenza di vetture riportata qui sotto, con un solo vincolo a bassa priorità (*LPI*) e tre ad alta (*HP1*, *HP2*, *HP3*) supponendo di considerare gli “*HP_N*” nell'ordine proposto in tabella.

Id	0	1	2	3	4	5	6	7	8	9
Colore	4	3	2	2	3	2	10	1	10	1
HP1	1	0	1	0	0	0	1	0	1	1
HP2	0	0	1	1	0	0	0	1	1	1
HP3	1	1	0	1	1	1	0	1	0	0
LP1	1	0	1	0	1	0	1	0	0	1

Figura 3: Esempio del contenuto di *matr_ridotta*

Dovendo considerare come primo obiettivo la minimizzazione del numero di violazioni sui vincoli di rapporto ad alta priorità e, per semplicità, come secondo quello sul numero minimo di cambi di colore, il valore iniziale del primo attributo, per ogni elemento dell'albero, sarà quello riportato nello schema sottostante.

5	5								
3	2	2	3						
0	3	0	2	1	1	3	0		

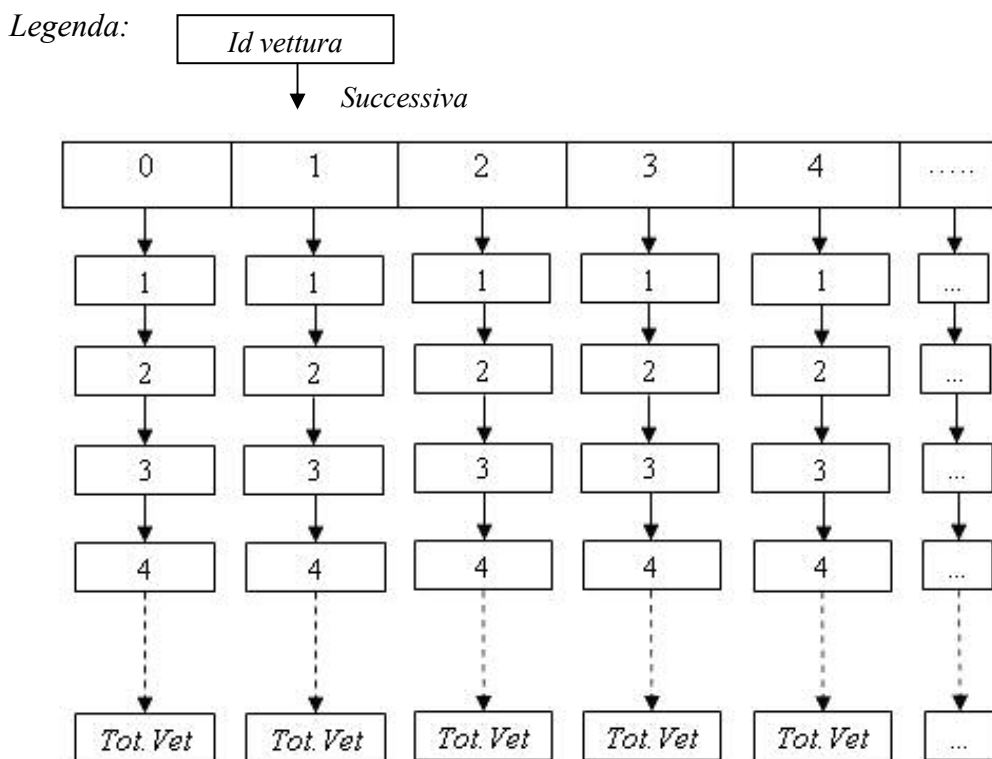
Figura 4: Esempio del valore del campo "quanti" della struttura Albero

I valori della prima riga dimostrano che, per quanto riguarda il primo vincolo considerato (HP1), esistono cinque vetture che non ne richiedono la presenza ed altre cinque che lo fanno. Quelli nella seconda riga che indicano che sono presenti tre vetture che possiedono la codifica binaria "00" nei primi due vincoli considerati, altre due con la codifica "01" e così via per tutti gli altri elementi.

Il valore dell'attributo che descrive quante di queste vetture hanno già una posizione fissa nella sequenza assume ovviamente valore iniziale zero.

Si procede quindi ad associare delle liste di vetture ad ogni posizione della sequenza. Inizialmente ogni posizione corrisponde a una lista contenente tutte le vetture.

La figura sottostante mostra l'insieme di liste generate ed associate ad ogni posizione della sequenza.



Struttura Dati 6: Insieme di liste associate ad ogni posizione della sequenza

In seguito si inizializzano le matrici utilizzate per la determinazione ed il salvataggio delle sequenze. A questo punto l'algoritmo possiede tutte le strutture per eseguire la fase successiva riguardante l'elaborazione dei dati.

In base alle caratteristiche dei veicoli, l'algoritmo stabilisce, per ogni posizione della sequenza finale di vetture, quale valore di ogni vincolo deve essere inserito per evitare che si verifichino delle violazioni.

L'operazione viene eseguita definendo i valori di ogni vincolo dalla prima all'ultima posizione della sequenza. Come prima operazione, la funzione controlla che nelle P-1 posizioni che precedono quella in esame, non vi siano già un

numero di imposizioni di tale limitazione pari al numero massimo di vetture associabili al vincolo di rapporto senza che sia introdotta alcuna violazione.

In questo caso, il procedimento implementato, prevede di confrontare la frequenza delle richieste di ogni singolo vincolo, all'interno della sequenza di vetture da produrre, con quella parziale degli "1" inseriti fino a alla posizione che precede quella attuale. La prima è calcolata in base al rapporto tra il valore del numero totale di richiesta dell'optional in esame per il numero totale di vetture da schedulare. La frequenza temporanea, invece, rapporta il numero di richieste di limitazioni alla fase di assemblaggio inserite sulla riga in esame fino a quel momento, con il numero di posizioni fino ad ora considerate.

Qualora la frequenza globale sia superiore a quella temporanea, l'algoritmo deciderà, di inserire una vettura che possieda la richiesta dell'optional in esame. Nel caso opposto invece, l'algoritmo richiederà per la specifica collocazione, una vettura che non preveda la presenza di quella limitazione nella fase di assemblaggio.

Qualora nelle P-1 posizioni siano già presenti il numero massimo di imposizioni perché non si verifichi la violazioni, il programma forzerà l'inserimento, nella posizione corrente, solo di quelle vetture che non necessitano di tale optional.

Deciso il valore che il vincolo corrente deve possedere nella posizione in esame, prima di confermarlo, è necessario accertarsi che esistano delle vetture con la caratteristica indicata, e che non siano già state tutte utilizzate nelle posizioni precedenti.

Per fare questo si accede ai dati contenuti nella struttura *Albero*: l'algoritmo si riferisce alla riga della struttura corrispondente al numero di vincoli fino ad ora esaminati.

Per identificare l'elemento, invece, converte da binario a decimale la combinazione di limitazioni sui vincoli associati alla posizione corrente. L'algoritmo si accerta che il numero di vetture, con quella codifica, sia maggiore di zero e che il numero di auto di quel tipo richieste in posizioni antecedenti non sia uguale a quelle totali.

Ad esempio, se si suppone di dover stabilire il valore relativo al secondo vincolo della sequenza di automobili ipotizzata in figura 3, l'algoritmo si trova a lavorare su una struttura provvisoria simile a quella sottostante:

Posizione	0	1	2	3	4	5	6	7	8	9
Colore	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
HP1	1	0	1	0	0	0	1	0	1	1
HP2	0	-	-1	-1	-1	-1	-1	-1	-1	-1
HP3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
LP1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Figura 5: Contenuto di `matr_help` durante l'elaborazione dei dati

dove le posizioni in cui il valore associato è “-1” rappresentano le decisioni che non sono ancora state prese in considerazione o, nel caso dei colori e di LP1, che non devono essere prese in considerazione a causa della priorità degli obiettivi. Supponiamo che l'algoritmo, dopo l'analisi delle frequenze, imponga il valore “1” al vincolo HP2 della posizione 1 (in figura è indicata con un tratteggio): prima di inserirlo definitivamente, l'algoritmo deve controllare la struttura Albero riportata in figura 4 ed in particolare l'elemento la cui codifica binaria corrisponde a “01” (le due limitazioni contornate da un rettangolo in tabella).

Naturalmente, se non esistono vetture con la combinazione voluta di attributi, sui vincoli esaminati, significa che sicuramente ne esistono che hanno l'ultimo elemento di valore opposto a quello richiesto dall'algoritmo e quindi nella posizione in esame viene forzato quest'ultimo valore. Questa forzatura, come comprensibile, può causare una violazione. Tutto ciò accade sicuramente se

nelle P posizioni che precedono quella in esame, ci sono già più di N vetture che impongono tale limitazione.

Il valore dell'elemento della colonna in esame, una volta deciso, viene salvato nella matrice che contiene le scelte effettuate, dopodichè si eliminano i riferimenti alle vetture presenti nella lista associata alla posizione in esame, che possiedono il valore opposto dell'attributo appena definito.

In questo modo, in ogni posizione della sequenza, è presente la lista delle sole vetture le cui caratteristiche, in termini di ratio constraints, corrispondono a quanto determinato dall'algoritmo.

Una volta considerati tutti i vincoli, per definire la sequenza di produzione, è necessario fissare, per ogni posizione, una sola tra le automobili ammissibili. La procedura che si occupa di questo si avvale dell'array denominato *Pos_colonne*, che definisce una soluzione, dato che per ogni posizione specifica l'indice dell'automobile ad essa associata.

Determinata l'automobile, l'algoritmo deve, naturalmente, eliminarla da tutte le altre liste, per evitare che possa essere inserita in più di una posizione. La dimensione delle liste, quindi, diminuisce man mano che si stabiliscono le posizioni di ogni vettura.

In generale l'algoritmo sceglie, per ogni posizione la prima vettura presente nella lista associata. Tuttavia, se il secondo obiettivo è minimizzare il numero di cambi di colore, si sceglie la prima vettura che ha lo stesso colore di quella presente nella posizione precedente; se la posizione da stabilire è la prima della sequenza, il colore di riferimento è quello dell'ultima vettura prodotta il giorno prima.

Definita la sequenza di vetture, il programma calcola il punteggio della soluzione generata seguendo le specifiche riportate nel capitolo 2 e, se questo

punteggio è il migliore ottenuto, ovvero minore di quello di tutte le altre soluzioni generate, il contenuto della sopraccitata matrice viene salvato nella struttura atta a contenere la miglior soluzione generata dalla tecnica costruttiva in assoluto.

4.3.2 Generazione della soluzione basata sul minimo numero di cambi di colore

Alcuni scenari prevedono, come primo obiettivo, di minimizzare il numero di lavaggi delle pistole, nel reparto di verniciatura dell'azienda. L'algoritmo che porta alla generazione di una soluzione, impostata sulle proprietà dei ratio constraints, è del tutto simile a quello precedentemente descritto, con l'aggiunta di alcune funzioni preliminari dedicate all'ottimizzazione delle tinte, come possibile vedere dal diagramma 5.

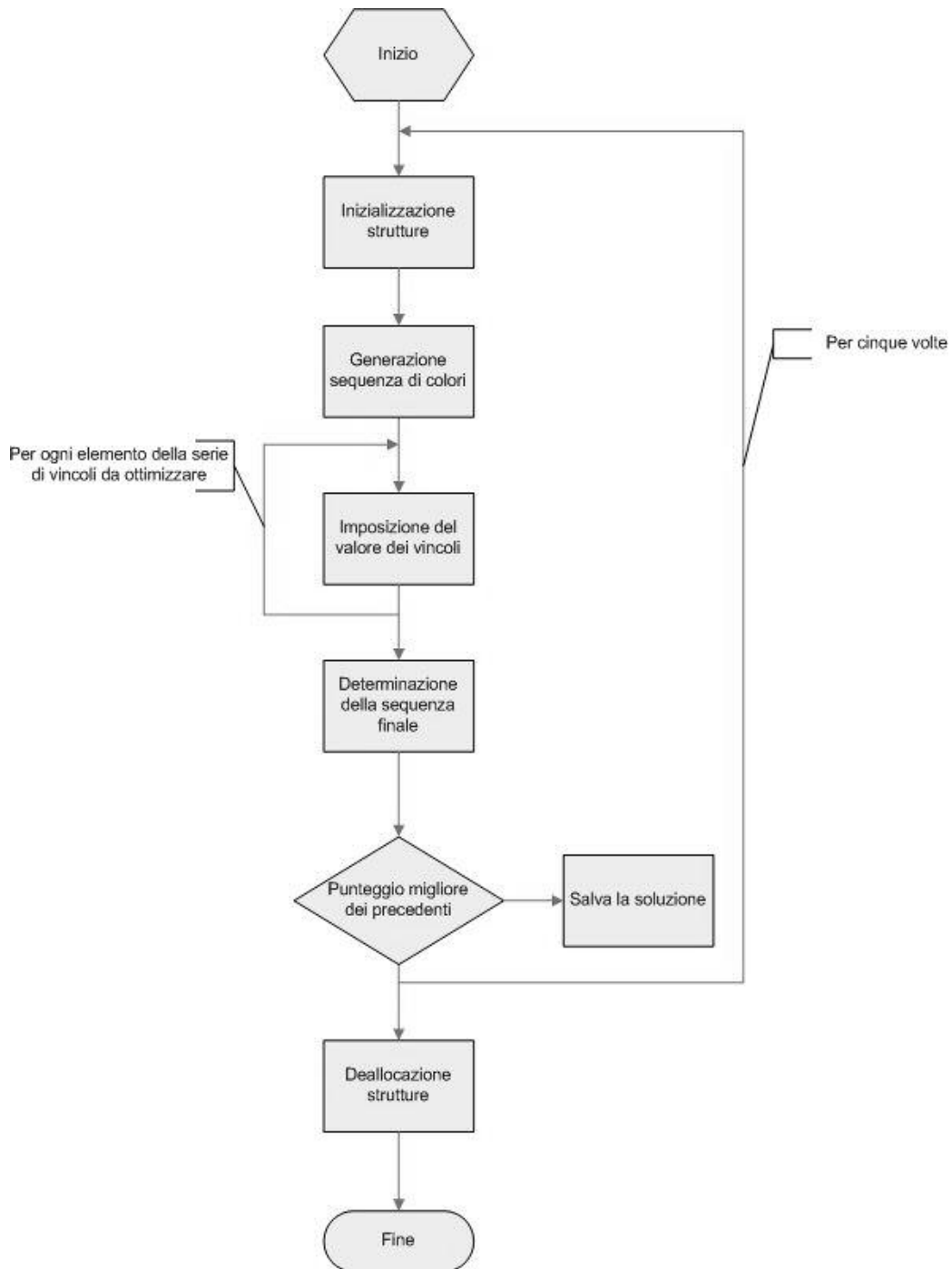


Diagramma 5: Soluzione con il minimo numero di cambi di colore

La prima delle funzioni sopraccitate, prevede di ordinare la sequenza di vetture da produrre in base alla tinta di colore richiesta per ogni automobile. Inizialmente, controlla qual'è il colore dell'ultima vettura prodotta nel giorno D-1. Fatto ciò viene cercato nell'insieme di automobili da produrre, se ne esiste almeno una che necessiti di essere verniciata di quel colore, per evitare in questo modo, di

avere una violazione sul cambio di colori già alla prima vettura inserita nella sequenza. In caso contrario, il primo colore preso in considerazione, sarà quello della prima vettura presente nella sequenza di automobili, caricata, come detto, in modo sequenziale dall'apposito file, per il giorno D.

Sempre riferendosi alla sequenza di vetture riportata in figura 3, dopo l'ordinamento, il contenuto della matrice di riferimento, considerando che l'ultima vettura del giorno D-1 sia di colore 3, corrisponde a quanto riportato in seguito.

Id	1	4	0	2	3	5	6	8	7	9
Colore	3	3	4	2	2	2	10	10	1	1
HPRC1	0	0	1	1	0	0	1	1	0	1
HPRC2	0	0	0	1	1	0	0	1	1	1
HPRC3	1	1	1	0	1	1	0	0	1	0
LPRC1	0	1	1	1	0	0	1	0	0	1

Figura 6: Contenuto di matr ridotta dopo l'ordinamento per colore

Ordinata la matrice in base al colore delle vetture, è necessario conteggiare quanti colori sono effettivamente presenti. Conoscendo questa quantità, infatti, l'algoritmo è in grado di istanziare l'array denominato *Tinte*, è composto da un numero di elementi pari alla quantità di colori differenti. Ogni cella del vettore corrisponde ad un diverso colore, presente nella sequenza di vetture e contiene diverse informazioni utili per le operazioni sulle tinte delle vetture. Queste informazioni sono elencate in seguito:

- *Colore*: Intero corrispondente al colore di verniciatura della vettura
- *Quanti*: Campo contenente la quantità di vetture da tinteggiare con quella tonalità

Per quanto riguarda tutti gli altri campi, il valore dipende in modo diretto dal valore restituito dal conteggio delle vetture, ovvero :

- *Blocchi*: Campo che rappresenta il numero di blocchi, di lunghezza *limit_brush*, di vetture dello stesso colore all'interno della sequenza finale.
- *Resto*: Resto della divisione tra il campo *Quanti* e l'attributo *limit_brush*
- *Mancanti*: Campo intero contenente il numero di blocchi, di quel colore, ancora da inserire nella sequenza di colori da generare.

Partendo dalla matrice ordinata, è facile prevedere che, il contenuto del vettore dedicato ai colori, al termine di questa fase, con un *limit_brush* pari a 2, possa essere il seguente:

Colore: 3	Colore: 4	Colore: 2	Colore: 10	Colore: 1
Quanti: 2	Quanti: 1	Quanti: 3	Quanti: 2	Quanti: 2
Blocchi: 1	Blocchi: 0	Blocchi: 1	Blocchi: 1	Blocchi: 1
Resto: 0	Resto: 1	Resto: 1	Resto: 0	Resto: 0
Mancanti: 1	Mancanti: 1	Mancanti: 2	Mancanti: 1	Mancanti : 1

Figura 7: Valore dei campi del vettore tinte

Definito il numero di blocchi per ogni colore, si crea una sequenza *random* di colori sulla quale l'algoritmo lavorerà per generare la soluzione finale.

Qualora sia presente, al giorno D, il colore dell'ultima vettura della sequenza del giorno D-1, è necessario porre un blocco di auto di questa tonalità come primo della sequenza, per evitare la violazione del vincolo sui colori, già a partire dalla prima vettura analizzata.

Supponendo, banalmente, di avere i medesimi dati dell'esempio precedente, alcune sequenze accettabili potrebbero essere :

3 – 2 – 10 – 2 – 1 – 4

3 – 4 – 10 – 1 – 2 – 2

3 – 2 – 2 – 1 – 10 – 4

eccetera...

Quanto detto per la procedura di generazione della sequenza di colori potrebbe essere implementato come segue.

```
Genera_sequenza_colori(int quanti_colori)
{
    Ultimo_ieri = Trova_ultimo_colore(matr_ieri);

    If (Ultimo_ieri_presente_in_matr_ridotta) then Sequenza[0] = Ultimo_ieri;
    Else Sequenza[0] = matr_ridotta[1][0];

    indice=1;

    While (indice < numero_colori_da_generare)
    {
        Tinta = Esegui_Random(); //Genera una cifra random tra 0 e numero colori
        Tinta = tinte [Tinta].colore;
        If (tinte [Tinta].mancanti>0)
        {
            Sequenza[indice] = tinta;
            indice = indice +1;
        }
    }
}
```

In seguito alla creazione della sequenza di colori, è opportuno riordinare l'insieme di vetture in modo tale che sia coerente con la sequenza di colori generata. Ogni colore specificato, prevede che vi sia, nella sequenza finale, una sottosequenza di al più *limit_brush* vetture di quella tinta.

Al termine di queste operazioni, il software prosegue applicando la procedura descritta in precedenza su ciascuna sottosequenza di vetture. Il procedimento è del tutto simile a quello descritto nel capitolo 4.3.1; cambiano le dimensioni ed il numero delle strutture dati in gioco.

La struttura *Albero* viene riprodotta tante volte, per ogni insieme di vetture di tinte differenti e le operazioni da eseguire considerano le automobili dello specifico colore.

Un'ulteriore differenza, è nell'inizializzare le liste di vetture inseribili nelle varie posizioni. Anche qui le vetture iniziali sono solo quelle del colore imposto dall'algoritmo per quella determinata posizione.

Si può facilmente comprendere che, nei diversi passi del programma, il numero di vetture da considerare volta per volta è notevolmente ridotto, rispetto alla soluzione che minimizza le violazioni sui ratio constraints. Proprio per questo l'algoritmo di cui stiamo parlando è decisamente più veloce rispetto al primo. Questo permette di poter ripetere tutte le operazioni finora descritte per differenti sequenze di colore e spiega la ripetizione visualizzata nel diagramma 5.

4.4 Generazione della soluzione per colonne

Una differente modalità di analisi delle vetture che compongono la sequenza di produzione del giorno D, consiste nel determinare, per ogni indice della serie, le caratteristiche ideali, in termini di ratio constraints e di colori, che un' automobile dovrebbe avere per una sequenza generata senza alcun tipo di violazioni. Anche in questo caso, però, l'algoritmo si adatta alle diverse esigenze delle aziende, a seconda che l'obiettivo a maggior priorità sia la minimizzazione delle violazioni sui ratio constraints alta priorità oppure quello del minimo numero di cambi di colore.

4.4.1 Generazione della soluzione basata sul minimo numero di violazioni sui ratio constraints ad alta priorità

In questa fase, le caratteristiche principali, analizzate dal programma per la scelta delle vetture, vertono fondamentalmente sui vincoli richiesti nella fase di assemblaggio. Il diagramma di flusso di questa fase è riportato in seguito.

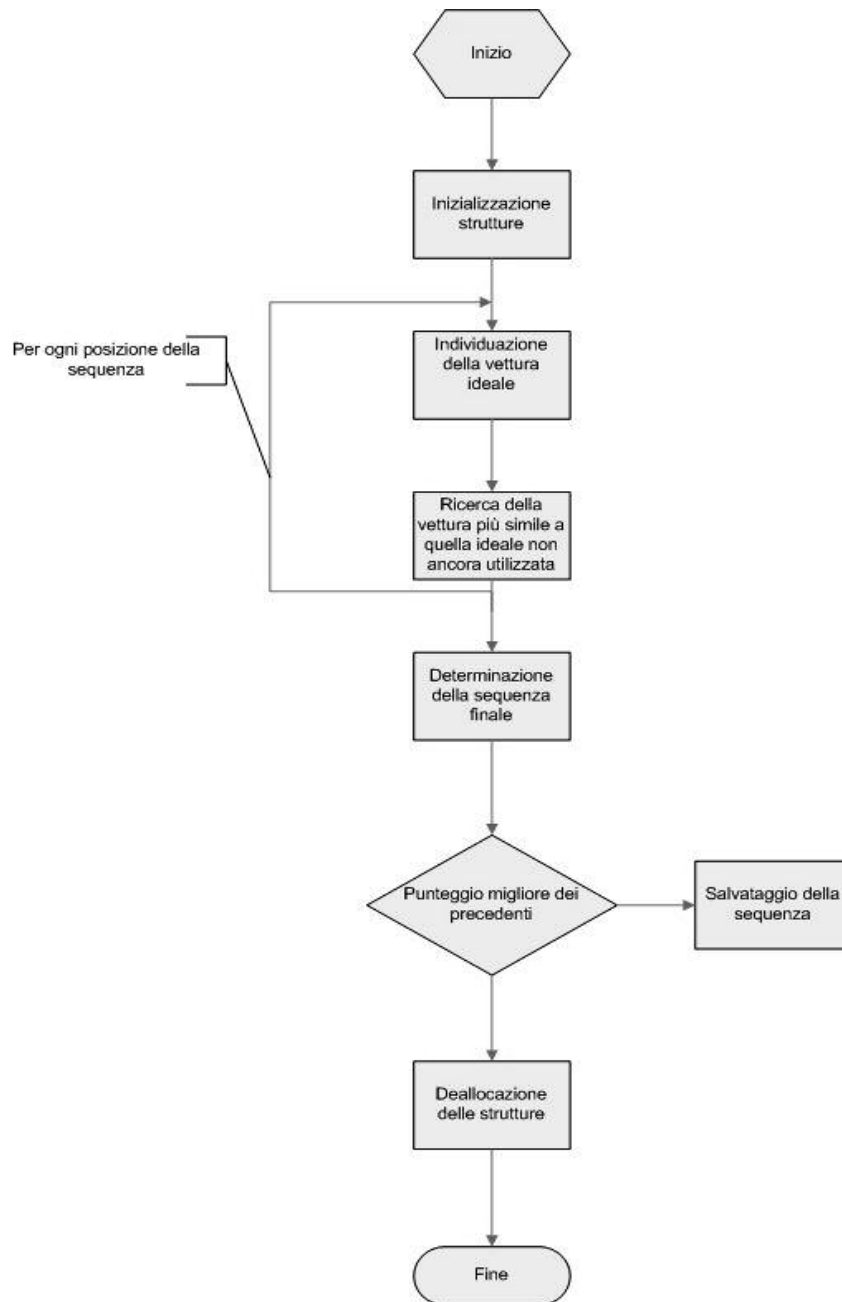


Diagramma 6: Soluzione con il minimo numero di violazioni su vincoli ad alta priorità

Quando l'obiettivo a massima priorità riguarda la minimizzazione del numero di violazioni sui ratio constraints l'algoritmo esegue le medesime operazioni di inizializzazione descritte nel capitolo 4.3.1.

Cambia, ovviamente, la modalità di selezione delle vetture da inserire in ogni posizione della sequenza. Per ogni posizione, attraverso un confronto fra le

densità globali e parziali, rispetto alla locazione in esame, vengono stabilite le imposizioni su ogni vincolo da considerare, con la medesima modalità descritta nel capitolo sopraccitato. Definendo i valori delle richieste di ogni optional da considerare, l'algoritmo stabilisce le caratteristiche della vettura ideale che dev'essere inserita in una posizione.

Anche in questo caso, i vincoli da considerare dipendono dalla priorità degli obiettivi; se il numero di violazioni dei vincoli a bassa priorità è il secondo requisito in ordine di importanza, il numero di decisioni da effettuare è pari al numero di ratio constraints totale. Nel caso contrario sono i soli vincoli ad alta priorità ad essere considerati.

Stabiliti i valori che una vettura deve possedere per poter essere inserita nella posizione in esame, senza causare alcuna violazione, l'algoritmo ricerca, tra tutte le automobili, che l'azienda deve produrre, quelle con le medesime caratteristiche richieste.

A tale scopo accede, come già descritto, alla struttura *Albero*, in particolare all'ultima riga, in quanto, questa volta, sono stati definiti i valori di tutti i vincoli da considerare. Se l'algoritmo deduce che esistono vetture con le caratteristiche specificate e non sono già state tutte scelte, un'automobile viene selezionata per essere inserita nella posizione corrente.

Esempio di sequenza ipotetica di vetture fornite:

Id	0	1	2	3	4	5	6	7	8	9
Colore	4	3	2	2	3	2	10	1	10	1
HP1	1	0	1	0	0	0	1	0	1	1
HP2	0	0	1	1	0	0	0	1	1	1
HP3	1	1	0	1	1	1	0	1	0	0
LP1	1	0	1	0	1	0	1	0	0	1

Figura 8: Esempio di una sequenza di vetture

Caratteristiche ideali stabilite dall'algoritmo per la prima soluzione della sequenza:

HP1	1
HP2	1
HP3	0
LP1	1

Figura 9: Caratteristiche ideali per la prima posizione

Vettura scelta dopo la scansione dell'insieme fornito:

Id	2
Colore	2
HP1	1
HP2	1
HP3	0
LP1	1

Figura 10: Vettura scelta dall'algoritmo

Qualora non esistano vetture che possiedono tutte le caratteristiche imposte dal programma, viene scelta quella che più si avvicina a quella ideale. Questa scelta è determinata attraverso un punteggio stabilito in base alle affinità tra le imposizioni sulle limitazioni stabilite dal software e quelle proprie di ogni vettura esaminata.

Il punteggio da associare a ciascuna vettura, viene calcolato sommando i seguenti valori:

- 1000 punti per ogni vincolo HP se questo è fissato a causa del raggiungimento, nell'intervallo in esame, del limite massimo di auto vincolate prima di una violazione.
- 100 punti per ogni vincolo HP in caso contrario

- 100 punti per ogni vincolo LP se questo è fissato a causa del raggiungimento, nell'intervallo in esame, del limite massimo di auto vincolate prima di una violazione.
- 10 punti per ogni vincolo LP in caso contrario

In questo modo, il punteggio associato ad ogni vettura, è determinato in base all'importanza dei vincoli uguali. Una limitazione imposta dall'algoritmo a causa del raggiungimento, nell'intervallo in esame, del limite massimo di auto vincolate prima di una violazione, è decisamente più importante di un vincolo imposto a causa di una differenza di densità; così come un vincolo di alta priorità necessita di maggior peso rispetto ad uno di bassa priorità.

Naturalmente, più di una vettura potrebbe possedere le caratteristiche di quella selezionata. Proprio per questo, si è deciso di non forzare l'inserimento in questa fase, ma solo di incrementare il campo contenente il numero di vetture di quel tipo all'interno della struttura *Albero* cosicché l'algoritmo può controllare se, una volta individuate le caratteristiche, esiste ancora la possibilità di inserirla nella posizione in esame. In caso contrario si procede utilizzando le vetture con il secondo miglior punteggio, e così via.

In questa fase non è da dimenticare, lo sfoltimento delle liste di automobili associate ad ogni posizione della sequenza dove, come già detto, vengono mantenute, le sole vetture che possiedono le caratteristiche più simili rispetto a quelle ideali.

Anche per la scelta delle vetture da fissare in ogni posizione, si ricorre a quanto detto in precedenza, ovvero, partendo dalla prima posizione della sequenza, si scandisce la lista di automobili associata ad ogni posizione. Qualora all'interno vi fosse, una vettura la cui carrozzeria dovrà essere verniciata con il medesimo colore di quella che la precede nella sequenza, o dell'ultima vettura del giorno $D - 1$ nel caso che la posizione in esame sia la prima, la scelta ricadrebbe su di questa. Questo comporta che la vettura in esame venga posta definitivamente

nella posizione stabilita, e quindi eliminata da tutte le liste che riferiscono alle diverse posizioni della sequenza.

La soluzione prodotta viene poi memorizzata nella struttura creata appositamente e valutata seguendo le priorità degli obiettivi. Anche qui, se il punteggio finale, basato sul numero di violazioni degli obiettivi, risulta inferiore rispetto a quello delle soluzioni generate precedentemente, prima della deallocazione delle strutture, si procede con l'aggiornamento del contenuto della matrice contenente la migliore soluzione generata.

4.4.2 Generazione della soluzione basata sul minimo numero di cambi di colore

La determinazione di una sequenza finale di autovetture da produrre, in questo caso, apporta modifiche rilevanti nelle fasi cruciali dell'algoritmo sopra descritto riportate nel diagramma sottostante.

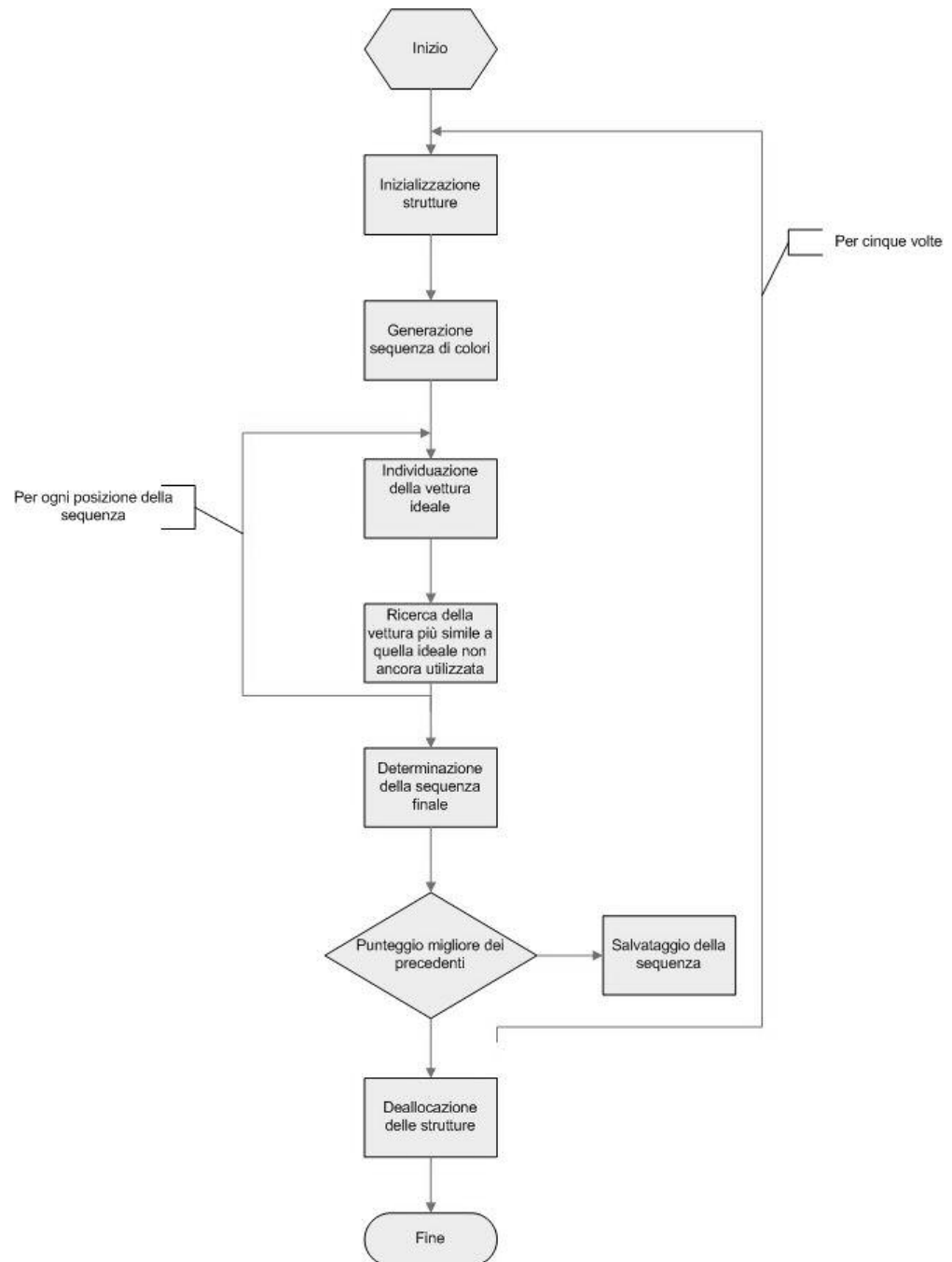


Diagramma 7: Soluzione con il minimo numero di cambi di colore

La sequenza di istruzioni, nella fase di inizializzazione, ricalca sostanzialmente, quanto descritto nel capitolo 4.3.2.

Alla fase iniziale, segue la generazione di una sequenza di colori, da rispettare nella generazione della sequenza finale, in base al contenuto del vettore

dedicato alle proprietà delle tinte presenti nella sequenza, esattamente come descritto nel capitolo sopraccitato.

L'obiettivo primario da considerare, infine, impone maggiori controlli nella determinazione delle vetture associabili ad ogni precisa posizione della sequenza finale. Qui, una volta stabilite le caratteristiche ideali che una vettura deve possedere per essere inserita in ogni posizione esaminata, la selezione delle vetture più simili a quanto stabilito dall'algoritmo viene effettuata esclusivamente sulle automobili da produrre che dovranno essere verniciate del colore definito in precedenza per la specifica posizione.

Le fasi successive, invece, rispecchiano quanto detto per l'implementazione basata sulla minimizzazione del numero di violazioni sui ratio constraints. Anche in questo caso, il numero inferiore di automobili da considerare volta per volta, accelera notevolmente l'elaborazione, consentendo di sperimentare molteplici sequenze di colori, ed aumentando perciò la possibilità di ottimizzare la soluzione finale.

4.5 Confronto tra generazione per righe e per colonne

Eseguendo l'algoritmo su tutti gli scenari, e controllando i punteggi delle sequenze restituite, si ottengono i risultati riportati in tabella, i quali ci consentono di trarre delle conclusioni in merito alla bontà degli algoritmi implementati:

N°	Scenario	Costo x righe	Costo x colonne	Obiettivi 1°/2°/3°
1	022_3_4_EP_RAF_ENP	18012	18012	HP/Colori/LP
3	024_38_3_EP_ENP_RAF	4274300	2133697	HP/LP/Colori
4	024_38_3_EP_RAF_ENP	4192094	2263384	HP/Colori/LP
5	024_38_5_EP_ENP_RAF	6274732	4063837	HP/LP/Colori
6	025_38_1_EP_ENP_RAF	1003384	195255	HP/LP/Colori
7	024_38_5_EP_RAF_ENP	6232512	4373097	HP/Colori/LP
8	025_38_1_EP_RAF_ENP	665665	64546	HP/Colori/LP
9	039_38_4_EP_RAF_ch1	1330300	1061100	HP/Colori
11	048_39_1_EP_ENP_RAF	649434	645218	HP/LP/Colori
12	048_39_1_EP_RAF_ENP	617124	517540	HP/Colori/LP
13	064_38_2_EP_RAF_ENP_ch1	227216	47518	HP/Colori/LP
14	064_38_2_EP_RAF_ENP_ch2	7720	7720	HP/Colori/LP
2	022_3_4_RAF_EP_ENP	114637	115305	Colori/HP/LP
10	039_38_4_RAF_EP_ch1	738100	740400	Colori/HP
15	064_38_2_RAF_EP_ENP_ch1	732292	782318	Colori/HP/LP
16	064_38_2_RAF_EP_ENP_ch2	310472	334057	Colori/HP/LP

Tabella 1: Tecniche costruttive a confronto

Com'è possibile notare dai dati, il confronto fra i due metodi implementati, offre due differenti responsi, a seconda che l'obiettivo a maggior priorità sia la minimizzazione le violazioni sui vincoli ad alta priorità, oppure del numero di cambi di colore.

Quando il primo obiettivo riguarda la minimizzazione del numero di violazioni sui ratio constraints ad alta priorità il secondo metodo implementato, surclassa, nella maggioranza dei casi il primo. La differenza media è del 37,3% (pari a 841.797,4 punti) come di seguito evidenziato nel grafico 1.

Nota:

Le colonne non visibili del grafico sono quelle relative ai risultati che riportano un punteggio molto inferiore rispetto a quelli riportati.

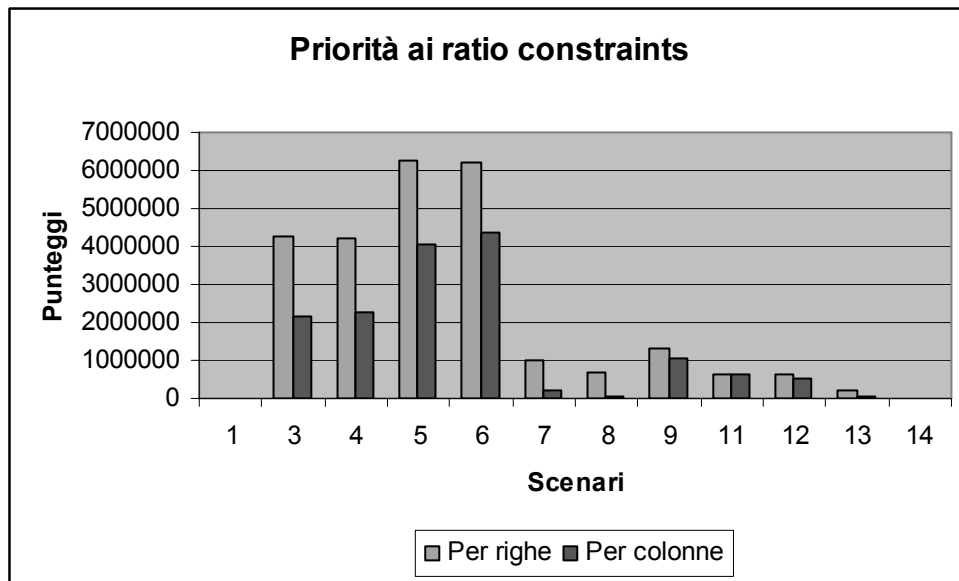


Grafico 1: Confronto degli algoritmi costruttivi sulla minimizzazione delle violazioni sui ratio constraints

Minimizzando il numero di cambi di colore, i punteggi migliori calcolati sulle sequenze di autovetture proposte dall'algoritmo, differiscono in media del 3,5 % (191.144,8 punti). Il metodo migliore risulta essere il primo implementato, anche se la differenza fra i due nella maggioranza dei casi è ridotta, come evidenziato anche dal grafico sottostante.

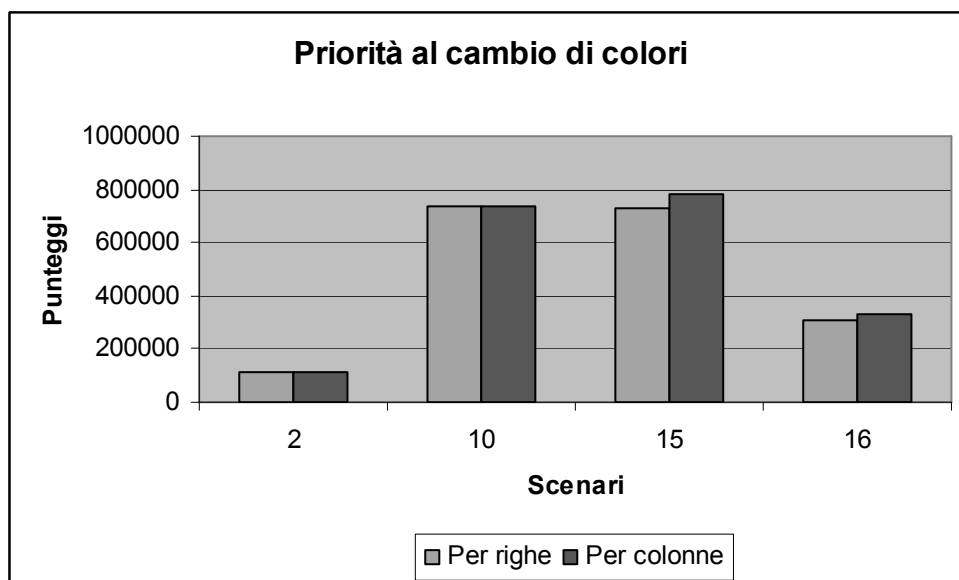


Grafico 2: Confronto degli algoritmi costruttivi sulla minimizzazione di cambi di colore

Il secondo metodo risulta migliore rispetto al primo, in quanto analizza tutti vincoli, decidendo in una sola volta, le caratteristiche fondamentali che le vetture devono possedere per essere adatte alla posizione in esame. Al contrario, il primo, decide tutti i valori che un vincolo deve assumere prima di passare ad analizzare il successivo. Questa scelta è penalizzante in quanto, le imposizioni sugli ultimi vincoli da considerare dovranno essere riscontrabili su un numero inferiore di vetture, e questo aumenta la probabilità che la scelta effettuata dall'algoritmo debba essere modificata in quanto nessuna delle vetture inseribili nella posizione la può rispettare. Il cambio del valore imposto dall'algoritmo comporta, nella maggioranza dei casi, il verificarsi di una violazione sul vincolo in esame.

5. Ricerca locale

La seconda parte del software è rappresentata da algoritmi di ricerca locale, applicati alla migliore soluzione ottenuta mediante tecnica costruttiva. La tecnica della ricerca locale appartiene alla categoria dei metodi euristici che non garantiscono di ottenere la soluzione ottima del problema, ma che in generale sono in grado di fornire una buona soluzione. La ricerca locale si basa su un'idea molto semplice: data una soluzione ammissibile, si esaminano le soluzioni ad essa vicine, in cerca di una soluzione migliore (tipicamente, con miglior valore della funzione obiettivo); se tale soluzione viene trovata essa diventa la nuova “soluzione corrente” ed il procedimento viene iterato, altrimenti, quando nessuna delle soluzioni vicine è migliore di quella “corrente”, l'algoritmo termina avendo determinata un ottimo locale. Il fattore che maggiormente influenza la ricerca locale è dato dalla soluzione iniziale, che può portare all'ottimo globale, ma tipicamente ad un ottimo locale. La miglior soluzione costruttiva generata viene perciò elaborata dall'algoritmo di ricerca locale.

La soluzione implementata si basa su due possibili “mosse”: lo scambio della posizione di due vetture e lo spostamento di un'auto in un'altra posizione.

Per effettuare queste operazioni si determinano i costi di scambio e spostamento per ogni possibile coppia di colonne, memorizzandoli in opportune matrici. I costi indicati sono in termini di variazione della funzione obiettivo (minimizzazione del numero di violazioni ai vincoli) pesando in modo differente i vincoli sulla verniciatura e quelli sulla linea di assemblaggio a seconda del particolare scenario che si presenta. Si possono pertanto avere valori di costo

positivi, che peggiorano la soluzione corrente, o negativi che invece portano un miglioramento.

Al termine del calcolo dei costi inizia l'algoritmo di ricerca locale vero e proprio che esamina, per ogni possibile vettura, tutti i possibili scambi, effettuando però solo quello, se presente, che determina il maggior miglioramento della soluzione corrente. Attuato lo scambio si aggiornano le strutture dati e si procede ad esaminare la vettura successiva. Il procedimento viene iterato finché l'algoritmo è in grado di determinare scambi miglioranti. Il passo successivo è invece rappresentato dal medesimo procedimento appena descritto, ma applicato agli spostamenti. L'intero insieme di scambi/spostamenti prosegue finché la soluzione non può più migliorare (si è giunti all'ottimo) oppure è scaduto il periodo di tempo massimo fissato per l'esecuzione.

La limitazione al tempo di esecuzione ha imposto di attuare solo gli scambi/spostamenti che, per ogni posizione, determinano il maggior miglioramento, anche se le colonne/posizioni coinvolte sono "lontane". Inoltre, sempre a causa del limite di tempo, gli aggiornamenti delle strutture dati sono stati mirati in modo tale da effettuare il minor numero possibile di calcoli necessari ed anche le funzioni di calcolo dei costi sono state definite per ottenere i valori desiderati sfruttando il meno possibile la CPU. Un altro accorgimento messo in pratica è stato quello di evitare di considerare l'ultimo degli obiettivi (verniciatura, linea di assemblaggio ad alta o bassa priorità a seconda dello scenario), permettendo di risparmiare tempo di calcolo pagando però un leggero peggioramento della soluzione, in quanto l'ultimo obiettivo è anche quello con peso inferiore. I risultati delle simulazioni, effettuate con i dati forniti, hanno evidenziato come il programma sia sempre in grado di migliorare in modo rilevante le soluzioni iniziali, fermo restando il limite dei 10 minuti al tempo di esecuzione. Sfruttando calcolatori più potenti o tempi di elaborazione più lunghi i risultati ottenuti sono stati ancora migliori.

6. Risultati

La Renault ha rilasciato tre differenti test set, dando la possibilità di ottimizzare il funzionamento del software, in particolare sono forniti:

- *Test set A*: disponibile sin dall'inizio del concorso. La giuria utilizza questi dati per la scelta dei finalisti. Questo è il test set di qualificazione.
- *Test set B*: fornito ai finalisti per mettere a punto i loro programmi per il test set X.
- *Test set X*: utilizzato per classificare i finalisti alla conferenza Roadef 2005. Questo è il test set finale che resta sconosciuto ai finalisti fino alla fine del concorso.

La messa a punto del programma dovrebbe migliorare la robustezza dell'algoritmo nei confronti di dati variabili. Questa robustezza è critica per applicazioni di schedulazione industriale. Non è immaginabile una messa a punto dei programmi ad ogni cambiamento nei dati di produzione (ratio constraint, caratteristiche dei veicoli, ecc.) in una qualsiasi fabbrica Renault.

6.1 Risultati del test set A

L'algoritmo è stato testato su tutti gli scenari appartenenti al Test set A, in quanto questi sono il mezzo attraverso il quale la giuria del concorso determinerà i concorrenti che avranno accesso alla fase finale.

I risultati sono riassunti in Tabella2:

N°	Nome dello Scenario	Violazioni r. c. ad alta priorità	Violazioni r. c. a bassa priorità	Cambi di colore	Punteggio Scenario
1	022_3_4_EP_RAF_ENP	0	1	105	10501
2	022_3_4_RAF_EP_ENP	48	4	11	114804
3	024_38_3_EP_ENP_RAF	54	84	543	548943
4	024_38_3_EP_RAF_ENP	38	258	366	416859
5	024_38_5_EP_ENP_RAF	69	82	518	698718
6	025_38_1_EP_ENP_RAF	0	383	863	39163
7	024_38_5_EP_RAF_ENP	63	124	422	672323
8	025_38_1_EP_RAF_ENP	0	1890	282	30090
9	039_38_4_EP_RAF_ch1	47	0	196	489600
10	039_38_4_RAF_EP_ch1	364	0	69	726400
11	048_39_1_EP_ENP_RAF	26	75	399	267899
12	048_39_1_EP_RAF_ENP	25	765	189	269665
13	064_38_2_EP_RAF_ENP_ch1	0	756	180	18756
14	064_38_2_EP_RAF_ENP_ch2	0	58	49	4958
15	064_38_2_RAF_EP_ENP_ch1	440	799	64	684799
16	064_38_2_RAF_EP_ENP_ch2	411	97	27	311197

Tabella 2: Risultati delle simulazioni

Per valutare le prestazioni del software riportiamo un confronto fra la soluzione corrispondente alla sequenza di vetture fornita in ingresso e la soluzione finale generata dal programma.

N°	Nome scenario	Punteggio Iniziale	Punteggio finale	Differenza	Obiettivi 1°/2°/3°
1	022_3_4_EP_RAF_ENP	27002	10501	61,11%	HP/Colori/LP
2	022_3_4_RAF_EP_ENP	114805	114804	0%	Colori/HP/LP
3	024_38_3_EP_ENP_RAF	828164	548943	33,72%	HP/LP/Colori
4	024_38_3_EP_RAF_ENP	766505	416859	45,62%	HP/Colori/LP
5	024_38_5_EP_ENP_RAF	1073792	698718	34,93%	HP/LP/Colori
6	025_38_1_EP_ENP_RAF	207790	39163	81,15%	HP/LP/Colori
7	024_38_5_EP_RAF_ENP	1026899	672323	34,53%	HP/Colori/LP
8	025_38_1_EP_RAF_ENP	31075	30090	3,17%	HP/Colori/LP
9	039_38_4_EP_RAF_ch1	1172900	489600	58,26%	HP/Colori
10	039_38_4_RAF_EP_ch1	729200	726400	0,38%	Colori/HP
11	048_39_1_EP_ENP_RAF	430134	267899	37,72%	HP/LP/Colori
12	048_39_1_EP_RAF_ENP	369061	269665	26,93%	HP/Colori/LP
13	064_38_2_EP_RAF_ENP_ch1	40287	18756	53,44%	HP/Colori/LP
14	064_38_2_EP_RAF_ENP_ch2	284650	4958	98,26%	HP/Colori/LP
15	064_38_2_RAF_EP_ENP_ch1	687043	684799	0,33%	Colori/HP/LP
16	064_38_2_RAF_EP_ENP_ch2	319761	311197	2,68%	Colori/HP/LP

Tabella 3: Differenza tra i risultati ottenuti e quelli iniziali

Differenza media = **35,76%** (175274,6 punti)

Il buon comportamento del programma è evidenziato dal fatto che tutte le soluzioni sono migliori di quelle fornite. Tuttavia si possono notare notevoli differenze nei risultati, in base alle priorità assunte dagli obiettivi come riportato nella tabella seguente:

Obiettivi	Differenza media
HP/Colori/LP	47,66%
Colori/HP/LP	0,85%
HP/LP/Colori	46,88%

Tabella 4: Differenza media a seconda degli obiettivi

Un riscontro dell'influenza dei dati sui risultati finali, lo possiamo ottenere analizzando gli scenari nei quali l'obiettivo a massima priorità è minimizzare il numero di cambi di colore. In questi casi, spesso, la soluzione iniziale minimizza, di per sé, il numero di lavaggi delle pistole, per cui il miglioramento ottenibile, non riduce in modo evidente il punteggio. Quanto detto in precedenza è chiaramente visibile nel grafico sotto riportato:

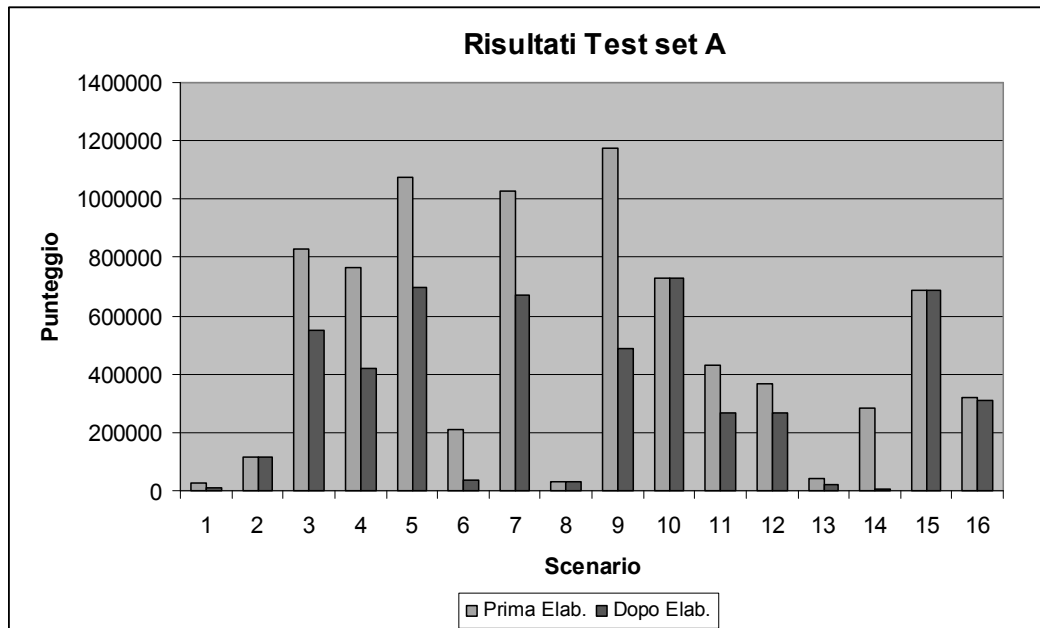


Grafico 1: Confronto dei risultati ottenuti

6.2 Gestione del limite di tempo

La limitazione del tempo di esecuzione ha imposto l'inserimento all'interno del software di funzioni appositamente dedicate a tale scopo. Queste funzioni, se da un lato assicurano il rispetto del limite, dall'altro determinano un inevitabile calo delle performance in quanto vengono richiamate frequentemente ed in diverse sezioni del software stesso. Questo perché, per alcune parti del codice, non è possibile prevedere a priori il tempo di esecuzione; ad esempio non è prevedibile sapere quanti scambi / spostamenti saranno effettuati sui dati relativi ad un qualsiasi scenario, e nemmeno quanti cicli di scambio / spostamento saranno necessari.

Quando viene raggiunto il limite dei 10 minuti di esecuzione il software interrompe l'ottimizzazione e genera in uscita la miglior soluzione trovata fino a quel momento, anche se questa non rappresenta la migliore possibile. Se,

viceversa, la miglior soluzione viene raggiunta prima della scadenza del limite di tempo, viene generato l'output e l'esecuzione termina.

Per compensare la “perdita” dovuta al controllo del tempo di esecuzione, e comunque per velocizzare il codice, si è pensato di utilizzare le funzionalità di ottimizzazione fornite dai compilatori. A tal proposito sono state effettuate differenti prove, compilando lo stesso codice sorgente, con diversi compilatori, ognuno dei quali è stato settato per fornire il codice eseguibile più veloce possibile. Sono stati testati i seguenti compilatori reperiti in Internet:

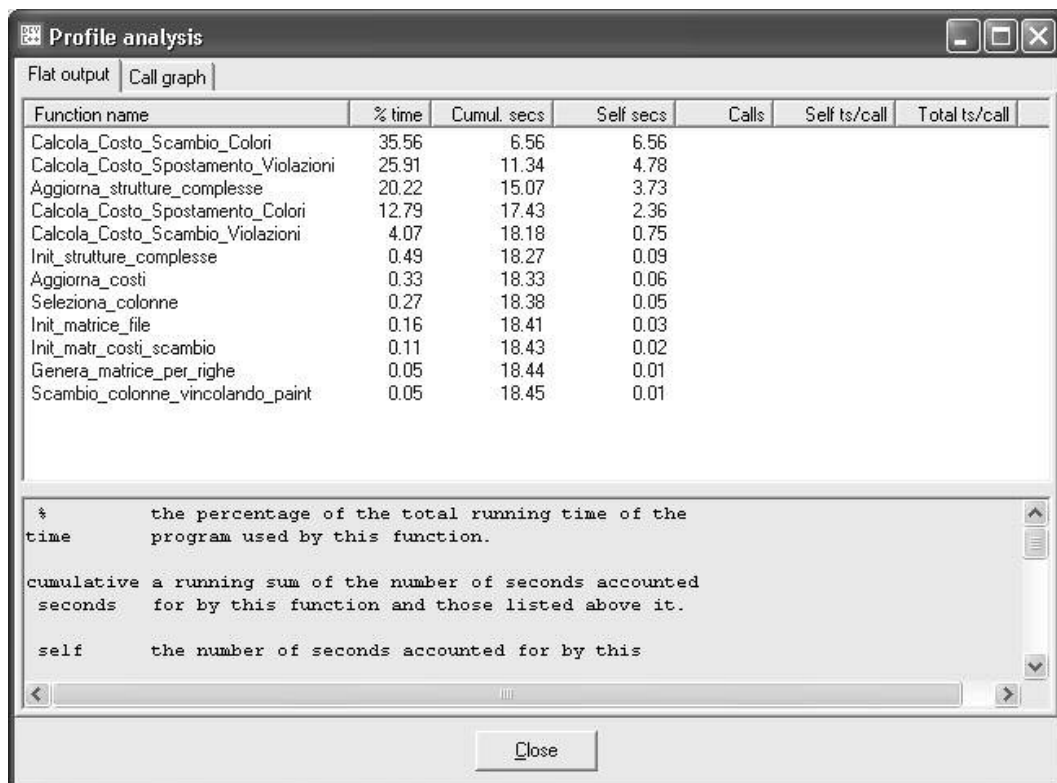
- GCC
- LCC
- TurboC
- DMC
- Borland C Compiler

Gli eseguibili ottenuti sono stati testati con il medesimo scenario ed il compilatore che ha fornito il minor tempo di esecuzione è stato GCC. Esso ha consentito di incrementare notevolmente la velocità di esecuzione nella misura di circa il 30% rispetto alla versione generata senza alcuna ottimizzazione.

6.3 Risultati del profiling

L'esecuzione del software ha un limite alla durata massima (10 minuti). Per questo si è resa utile l'attività di *profiling*, che permette di conoscere quali sono le procedure più utilizzate, mostrando per ognuna di queste il tempo di CPU utilizzato, nonché la percentuale del tempo di esecuzione occupato rispetto al totale.

I risultati ottenuti, hanno mostrato come l'algoritmo si adatti all'ordine degli obiettivi di ottimizzazione. Infatti, la graduatoria delle procedure maggiormente richiamate, varia di scenario in scenario. Ad esempio, nello scenario "064_38_2_RAF_EP_ENP_CHI", che come obiettivo primario richiede la minimizzazione dei cambi di colore, l'analisi ha fornito il seguente risultato:



The screenshot shows a window titled "Profile analysis" with two tabs: "Flat output" (selected) and "Call graph". Below the tabs is a table with the following data:

Function name	% time	Cumul. secs	Self secs	Calls	Self ts/call	Total ts/call
Calcola_Costo_Scambio_Colori	35.56	6.56	6.56			
Calcola_Costo_Spostamento_Violazioni	25.91	11.34	4.78			
Aggiorna_strutture_complesse	20.22	15.07	3.73			
Calcola_Costo_Spostamento_Colori	12.79	17.43	2.36			
Calcola_Costo_Scambio_Violazioni	4.07	18.18	0.75			
Init_strutture_complesse	0.49	18.27	0.09			
Aggiorna_costi	0.33	18.33	0.06			
Seleziona_colonne	0.27	18.38	0.05			
Init_matrice_file	0.16	18.41	0.03			
Init_matr_costi_scambio	0.11	18.43	0.02			
Genera_matrice_per_righe	0.05	18.44	0.01			
Scambio_colonne_vincolando_paint	0.05	18.45	0.01			

Below the table is a legend:

- % time: the percentage of the total running time of the program used by this function.
- cumulative seconds: a running sum of the number of seconds accounted for by this function and those listed above it.
- self: the number of seconds accounted for by this

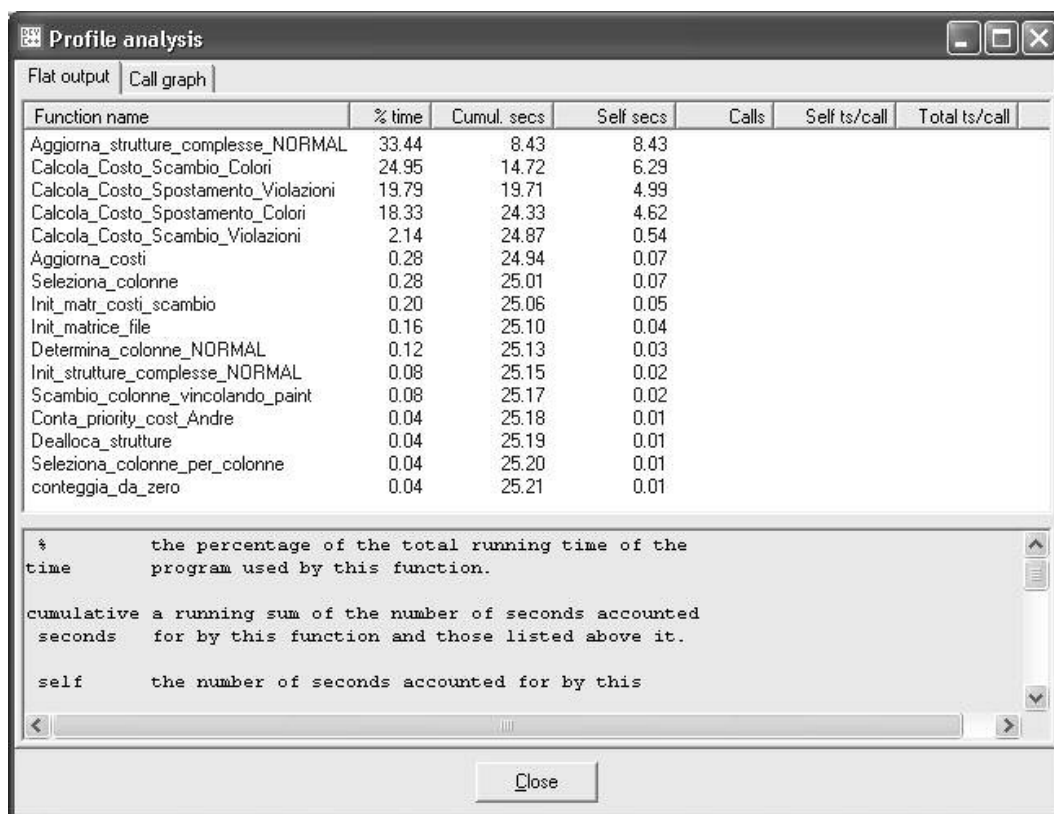
At the bottom of the window is a "Close" button.

Profiling 1: Scenario 064_38_2_RAF_EP_ENP_CHI

Dallo *screenshot* si evince che il maggior peso ricade sull'algoritmo di ricerca locale; infatti, le tecniche costruttive basate sui colori hanno una durata inferiore perché il numero di vetture esaminato in ogni posizione è ridotto alle sole auto del colore in esame.

Passando allo scenario in cui l'obiettivo più importante è quello sui ratio constraints ad alta priorità, la distribuzione dei tempi di CPU è differente. In questo caso la ricerca locale occupa ancora la maggior parte del tempo di elaborazione ma, al contrario dell'esempio precedente, la procedura più onerosa

appartiene alla tecnica costruttiva. Il riscontro è evidenziato nei dati appartenenti allo scenario “022_3_4_EP_RAF_ENP” riportato in seguito:



The screenshot shows a window titled "Profile analysis" with two tabs: "Flat output" and "Call graph". The "Flat output" tab is active, displaying a table with the following data:

Function name	% time	Cumul. secs	Self secs	Calls	Self ts/call	Total ts/call
Aggiorna_strutture_complesse_NORMAL	33.44	8.43	8.43			
Calcola_Costo_Scambio_Colori	24.95	14.72	6.29			
Calcola_Costo_Spostamento_Violazioni	19.79	19.71	4.99			
Calcola_Costo_Spostamento_Colori	18.33	24.33	4.62			
Calcola_Costo_Scambio_Violazioni	2.14	24.87	0.54			
Aggiorna_costi	0.28	24.94	0.07			
Seleziona_colonne	0.28	25.01	0.07			
Init_matr_costi_scambio	0.20	25.06	0.05			
Init_matrice_file	0.16	25.10	0.04			
Determina_colonne_NORMAL	0.12	25.13	0.03			
Init_strutture_complesse_NORMAL	0.08	25.15	0.02			
Scambio_colonne_vincolando_paint	0.08	25.17	0.02			
Conta_priority_cost_Andre	0.04	25.18	0.01			
Dealloca_strutture	0.04	25.19	0.01			
Seleziona_colonne_per_colonne	0.04	25.20	0.01			
conteggia_da_zero	0.04	25.21	0.01			

Below the table, there is a legend:

- % time: the percentage of the total running time of the program used by this function.
- cumulative seconds: a running sum of the number of seconds accounted for by this function and those listed above it.
- self: the number of seconds accounted for by this

At the bottom of the window is a "Close" button.

Profiling 2: Scenario 022_3_4_EP_RAF_ENP

Come ultimo esempio prendiamo in considerazione lo scenario “025_38_1_EP_ENP_RAF” in cui le variazioni di colore risultano l’ultimo obiettivo in ordine di importanza. L’analisi di profiling evidenzia un’elevato peso del tempo di esecuzione di due procedure.

La prima è quella relativa al calcolo dei costi; la minimizzazione dei cambi di colore è l’ultimo obiettivo da considerare perché l’algoritmo è costretto a considerare tutti i ratio constraints, anche quelli a bassa priorità. La seconda riguarda l’inizializzazione della struttura *Albero* utilizzata dalla tecnica costruttiva. Una fase di inizializzazione così lenta è dovuta al fatto che sono presenti nello scenario un elevato numero di vincoli, tutti da considerare, e questo aumenta in maniera esponenziale le dimensioni di questa struttura dati.

Profile analysis

Flat output | Call graph

Function name	% time	Cumul. secs	Self secs	Calls	Self ts/call	Total ts/call
Calcola_Costo_Scambio_Violazioni	62.74	185.06	185.06			
Azzeramento_Albero	25.63	260.66	75.60			
Aggiornamento_costi	5.13	275.78	15.12			
Aggiornamento_strutture_complesse_NORMAL	3.09	284.89	9.11			
Riempi_Albero	1.12	288.19	3.30			
Crea_Albero	0.99	291.11	2.92			
Genera_matrice_per_colonne_NORMAL	0.29	291.98	0.87			
Seleziona_colonne	0.26	292.75	0.77			
Init_matrice_help	0.16	293.22	0.47			
pow	0.08	293.45	0.23			
Genera_matrice_per_righe_NORMAL	0.07	293.65	0.20			
Seleziona_colonne_per_colonne	0.07	293.85	0.20			
conteggia_da_zero	0.05	294.01	0.16			
Init_matr_costi_scambio	0.04	294.13	0.12			
Crea_file_soluzione	0.03	294.22	0.09			
Crea_matrice	0.03	294.31	0.09			
conteggia	0.03	294.39	0.08			
Converti_in_decimale	0.02	294.46	0.07			
Scambia_colonna	0.02	294.52	0.06			
main	0.02	294.58	0.06			
Determina_colonne_NORMAL	0.02	294.63	0.05			

% the percentage of the total running time of the
 time program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds for by this function and those listed above it.

self the number of seconds accounted for by this

Close

Profiling 3: Scenario 025_38_1_EP_ENP_RAF

In generale l'analisi di profiling conferma i risultati attesi, ossia la tecnica greedy è in grado di fornire una soluzione in tempi ragionevoli, a discapito però dell'ottimalità di quest'ultima. Quanto prodotto dall'algoritmo costruttivo viene in seguito ottimizzato mediante la ricerca locale che risulta la parte più onerosa in termini di tempo, anche perché a priori non è possibile conoscere la durata delle elaborazioni che verranno effettuate.

7. Possibili miglioramenti

L'algoritmo è già in grado di fornire sequenze di vetture che rappresentano buone soluzioni per il problema trattato. Nonostante ciò, sono stati individuati aspetti ai quali apportare delle modifiche per migliorare la qualità dei risultati proposti dall'algoritmo greedy ed incrementare la velocità di esecuzione della ricerca locale.

7.1 Migliorare la qualità dei risultati

Le tecniche greedy forniscono delle soluzioni in tempo breve ma, purtroppo, queste non sempre risultano di buona qualità a causa dei meccanismi di decisione e di pesatura delle automobili da inserire in ogni posizione della sequenza. Migliorando i criteri di selezione e specificando ulteriormente le condizioni di scelta, mediante un miglior utilizzo delle informazioni relative a frequenza e/o densità dei ratio constraints, si potrebbe ottenere una soluzione che si avvicini maggiormente all'ottimo globale. Questo faciliterebbe i compiti della ricerca locale oltre a ridurre i tempi di esecuzione.

7.2 Incrementare la velocità di esecuzione

L'attività di profiling ha evidenziato come le procedure che richiedono maggior tempo di CPU, siano quelle relative al calcolo dei costi, sia di scambio che di spostamento, e riferite sia ai ratio constraints che ai cambi di colore. Queste procedure sono già state implementate attraverso metodi che garantiscono una computazione rapida, in particolare per quel che riguarda i ratio constraints, tuttavia si potrebbero trovare soluzioni algoritmiche in grado di eseguire la computazione in un tempo inferiore.

7.3 Miglioramenti generali

Entrambe le tecniche implementate hanno la caratteristica di non considerare come fondamentale l'ottimizzazione dell'ultimo obiettivo; quindi un miglioramento generale si avrebbe valutando in modo approfondito tutti gli obiettivi. Questo probabilmente andrà a discapito del tempo di elaborazione, ma, lo studio di tecniche algoritmiche più raffinate e senza limiti al tempo di esecuzione, le soluzioni a cui si potrebbe giungere sarebbero senz'altro migliori di quelle attuali.

A handwritten signature in dark ink, appearing to read 'D. B...', is located at the bottom right of the page.

Bibliografia

1. Roadef Challenge 2005 Problem Description – [*Capitoli 1, 2, 3, 6*]
2. Algoritmi e strutture Dati - *Alan Bertossi* - Utet 2001 [*Tecniche Costruttive - Introduzione*]
3. Introduzione agli algoritmi vol. 2 – *T.H. Cormen, C.E. Leiserson, R.L. Rivest – Jackson Libri 1996* [*Tecniche Costruttive - Introduzione*]