La Sintesi ad Alto Livello: Estensioni al Force Directed Scheduling

26 luglio 2006

Marco Lattuada

Relatore:
Prof. Fabrizio Ferrandi

Obiettivo

Modificare il Force Directed Scheduling proposto da Paulin e Knight:

- migliorando la qualità dei risultati;
- diminuendo la complessità temporale dell'algoritmo;
- estendendo il campo di applicabilità dell'algoritmo aggiungendo la possibilità di:
  - introdurre vincoli sul numero di unità funzionali;
  - eseguire il binding sul tipo di risorsa.

La Sintesi ad Alto Livello

- Sintesi ad Alto Livello: trasformazione di una descrizione comportamentale del sistema in una descrizione a livello RT;
- Allocazione, Scheduling e Binding sono alcune delle sue fasi;
- queste tre fasi possono essere contemporanee o successive;
- Force Directed Originale: Allocazione + Scheduling
- Force Directed Proposto: Allocazione + Scheduling + Binding (parziale)

Allocazione - Scheduling - Binding

Allocazione

Vengono definite numero e tipo delle risorse: funzionali, di memorizzazione e di comunicazione.

Scheduling

Il tempo di esecuzione della funzionalità viene suddiviso in intervalli uguali chiamati passi di controllo. Le singole operazioni vengono assegnate ai singoli passi di controllo.

Binding

Gli elementi presenti nella descrizione di sistema vengono mappati su componenti architetturali (unità funzionali, bus, registri).

Il Force Directed Scheduling originale

- Algoritmo proposto da P. G. Paulin e J. P. Knight nel 1987;
- fissato il tempo di esecuzione minimizza il numero di risorse;
- cerca di distribuire uniformemente le operazioni:
  - associa ad ogni coppia <operazione-passo di controllo> una forza;
  - la forza è un numero razionale indice degli effetti dell'assegnamento;
  - ad ogni iterazione esegue l'assegnamento con forza minore (cioè migliore).

Dati del Problema

- SDG (*System Dependence Graph*) : grafo con sia dipendenze-dato sia dipendenze di controllo;
- Scheduling prodotto da ASAP e ALAP, finestra temporale e mobilità di ogni operazione;
- caratteristiche delle unità funzionali:
  - tipo di operazioni eseguibili;
  - tempo di esecuzione;
  - presenza di pipeline;
  - numero di unità (in caso di presenza di vincoli).

Formula per il calcolo della forza

- Il termine forza deriva dall'analogia con la legge di Hooke:

$$F = -kx$$

- ad ogni coppia <operazione-passo di controllo> è assegnata una probabilità: $prob(o,s) = \frac{1}{mobilita'}$
- $FORCE(o,s) = self\_force(o,s) + pred\&succ\_force(o,s)$
- $self\_force(o,s) = force(o,s)$
- $pred\&succ\_force(o,s) = \sum_{q \epsilon pred(o)} force(q,s) + \sum_{q \epsilon succ(o)} force(q,s)$
- $force(o,s) = \sum_{p \epsilon wind(o)} contrib(o,p)$
- $contrib(o,p) = DG(type(o),p) \cdot \triangle prob(o,p)$

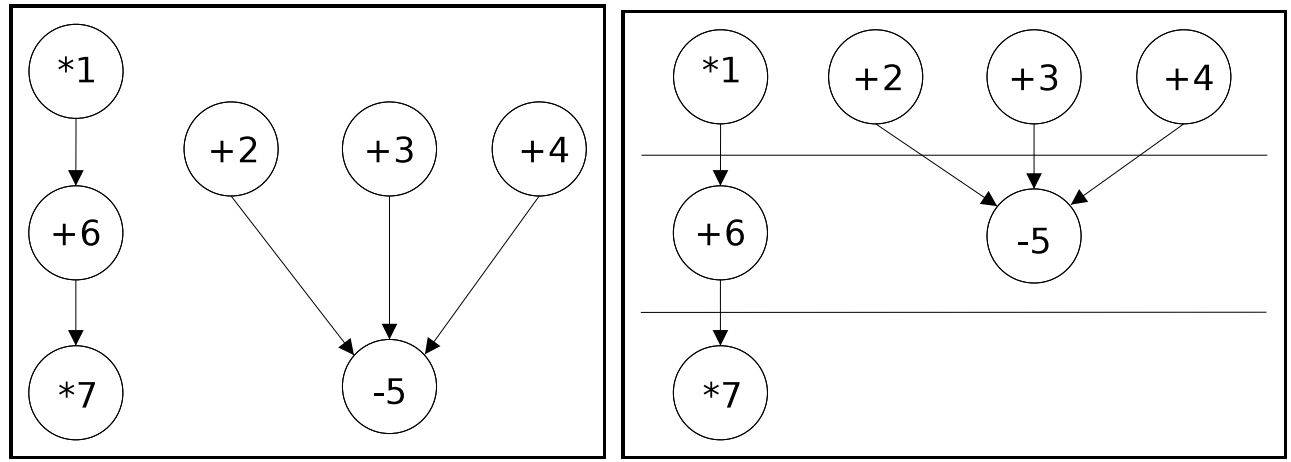
Estensioni alla versione base

Paulin e Knight hanno proposto delle estensioni che permettono di considerare:

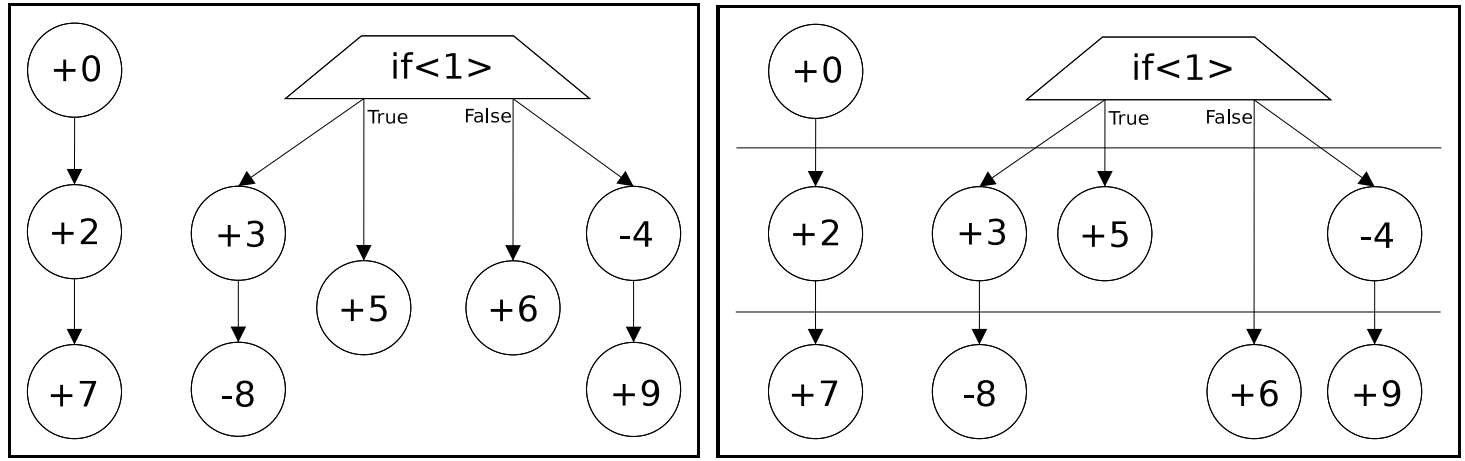
- costrutti ciclici e costrutti condizionali;
- minimizzazione di bus e registri;
- costo delle diverse unità funzionali;
- scheduling con chaining, con operazioni multiciclo, con unità funzionali dotate di pipeline;
- presenza di vincoli sul numero di risorse (escludendo vincoli temporali);
- presenza di vincoli temporali locali.

Limiti insiti nel Force Directed

- Presenza di più predecessori/successori dello stesso tipo



- Presenza di costrutti condizionali



Modifiche applicate:

- formula per il calcolo della forza;
- riduzione del peso relativo alle forze negative di predecessori e successori;
- criterio di scelta del prossimo assegnamento da effettuare;
- riduzione della complessità temporale dell'algoritmo tramite scheduling a blocchi;
- utilizzo contemporaneo di vincoli sul numero di risorse e vincoli temporali;
- binding sul tipo di unità funzionale.

Modifiche al Calcolo delle forze

Modifica al calcolo dei singoli contributi

- Il Force Directed Originale considera solo parzialmente le mutue esclusioni;
- ipotizza che ad ogni assegnamento corrisponda l'allocazione di una nuova unità funzionale in quel passo di controllo;
- l'unità potrebbe già essere disponibile perchè utilizzata da un'operazione in mutua esclusione;
- modifica effettuata: si sostituisce alla variazione di probabilità la variazione del numero medio di risorse necessarie allocate.

Modifica al peso delle forze di predecessori e successori

- Le forze di predecessori e successori sono approssimate;
- ad una forza negativa potrebbe corrispondere un falso miglioramento;
- modifica effettuata: si cerca di trascurare le forze di predecessori e successori negative diminuendone il peso.

Scelta del prossimo assegnamento

- Il vero problema dello scheduling consiste nello scegliere in che passo di controllo assegnare le singole operazioni;
- le forze di un'operazione forniscono un ordine di preferenza degli assegnamenti di quella operazione;
- se la differenza fra le forze è minima l'ordine può essere sbagliato;
- nuovo criterio di scelta: operazione con differenza maggiore fra forza migliore e forza media:
  - maggiore differenza $\Rightarrow$ minor peso dell'errore $\Rightarrow$ minor possibilità di fare una scelta svantaggiosa;
  - le forze simili possono differenziarsi nelle iterazioni successive.

Riduzione della complessita' dell'algoritmo

- Complessità originale dell'algoritmo: $O(c^2n^3)$
- complessità elevata per la visione globale del problema;
- scomposizione del problema in parti: scheduling di singole parti del sistema;
- durante lo scheduling di una parte si considerano le informazioni relative a tutto il sistema;
- ad ogni iterazione è necessario ricalcolare solo i dati della singola parte;
- per la scomposizione in parti si utilizzano i blocchi basici:
  - nuova complessità: $O(\frac{1}{B}c^2n^3)$;
  - nessun guadagno in assenza di costrutti condizionali.

Utilizzo contemporaneo di due tipi di vincoli

- Gli algoritmi di scheduling a tempo di esecuzione vincolato solitamente non consentono di vincolare il numero delle risorse;
- il numero delle risorse è la funzione obiettivo da minimizzare;
- nell'algoritmo proposto è possibile utilizzare entrambi i tipi di vincoli;
- le motivazioni sono
  - individuare unità non utilizzate;
  - minimizzare il numero di risorse non vincolate;
  - minimizzare il numero di interconnessioni o di registri;
  - uniformare l'utilizzo delle risorse nei diversi passi di controllo.

Aggiunta dei vincoli sulle risorse

Le modifiche effettuate per consentire l'aggiunta dei vincoli sulle risorse sono:

- aggiunta dei pesi alle somme di probabilità e quindi alle forze: il peso è pari all'inverso del numero di unità funzionali disponibili;
- introduzione del meccanismo di backtracking :
  - l'algoritmo può produrre una soluzione non accettabile perchè viola un vincolo;
  - uno o più assegnamenti effettuati vengono cancellati per riottenere una soluzione parziale accettabile;
  - l'algoritmo esegue nuovi assegnamenti per costruire una soluzione accettabile.

Albero di ricerca e tagli

- lo scheduling con vincoli può essere modellizzato come esplorazione di un albero di ricerca:
  - ogni nodo corrisponde ad una soluzione parziale;
  - ogni arco corrisponde ad un assegnamento;
- è possibile ridurre il tempo di ricerca di una soluzione accettabile tramite:
  - il taglio di sottoalberi il cui nodo radice corrisponda ad una soluzione esplicitamente o implicitamente non accettabile;
  - l'anticipazione di assegnamenti obbligatori;
  - backtracking anticipato;
  - riduzione del grado dell'albero di ricerca.

Binding sul tipo di unita' funzionale

- E' stata aggiunta la funzione di binding di un'operazione su un tipo di unità funzionale;
- si associa una forza ad ogni terna <operazione-passo di controllo-tipo di unità funzionale> ;
- $FORCE(o,s,t) = self\_force(o,s,t) + pred\&succ\_force(o,s,t) + assign\_force(o,s,t)$
- al concetto di probabilità di un'operazione di essere schedulata in un passo di controllo si sostituisce quello di percentuale di occupazione di un tipo di unità funzionale in un passo di controllo da parte di un'operazione ;
- in caso di mancanza di binding la nuova formulazione coincide con quella originale;
- è possibile utilizzare questa formulazione per tutte le casistiche del problema di scheduling.

Dimensioni dei benchmark

Benchmark	nOp	n° salti
Adpcm_decode	68	11
Adpcm_encode	83	14
Arf	34	0
Bandpass	52	0
Chemical	38	0
Dct	58	0
Dct_wang	57	0
Diffeq	27	1
Kim	34	2
Maha	30	5

Risultati Sperimentali vs FD Originale

nFu indica il numero complessivo di risorse allocate: l'algoritmo non discrimina fra i diversi tipi di unità non avendo informazioni relative ai costi

Benchmark	FD Orig.		FD Modif1.	
	nFU	tempo	nFU	tempo
Adpcm_decode	27	2.89	18	10.69
Adpcm_encode	27	5.84	19	43.68
Arf	14	0.23	8	0.25
Bandpass	20	0.55	11	0.57
Chemical	25	0.45	15	0.69
Dct	35	0.80	20	1.03
Dct_wang	29	0.65	20	0.85
Diffeq	14	0.09	8	0.29

Risultati Sperimentali: Complessita'

Riduzione del tempo di computazione a seguito della riduzione di complessità

Benchmark	FD Modif1.		FD Modif2.	
	nFU	tempo	nFU	tempo
Adpcm_decode	18	11.77	18	3.54
Adpcm_encode	19	43.30	19	7.92
Diffeq	8	0.28	8	0.25
Kim	6	0.68	6	0.58
Maha	5	0.41	5	0.25

Risultati Sperimentali vs List-Based

Guadagno in termini di unità funzionali allocate vincolate

		Kim		Maha	
Unità Funzionale	Vincolo	LB	FD	LB	FD
CMP	2			2	1
indirect_ref	inf	1	2	1	1
MINUS	2	2	1	2	1
PLUS	2	2	2	1	1
READ_COND	inf	1	1	2	1
Tempo computazione		0.11	0.18	0.14	0.12

Risultati Sperimentali vs List-Based

Guadagno in termini di unità funzionali allocate non vincolate

		Bandpass		Diffeq	
Unità Funzionale	Vincolo	LB	FD	LB	FD
ASSIGN	inf			1	1
CMP	2			1	1
indirect_ref	inf	13	6	7	3
MINUS	2	2	2	1	1
MUL	1	1	1	1	1
PLUS	2	1	2	1	1
READ_COND	inf			1	1
Tempo computazione		0.22	75.77	0.10	0.99

Conclusioni e Futuri Sviluppi

Conclusioni

La versione proposta dell'algoritmo:

- ha complessità inferiore a quella originale;
- fornisce risultati qualitativamente migliori di quella originale;
- permette l'introduzione di vincoli sul numero delle risorse;
- svolge anche il binding delle operazioni sui tipi di unità funzionali.

Futuri Sviluppi:

- ridurre la complessità in caso di assenza di costrutti di controllo;
- minimizzazione del costo di registri e interconnessioni;
- inserire informazioni relative al costo delle unità funzionali;
- valutare il guadagno dovuto ad un calcolo più preciso delle forze;
- ridurre il tempo di computazione a scapito della memoria utilizzata.