Steepest Descent Approaches for the Minimization of Open Stacks

Júnior Rhis
Marco Antonio M. Carvalho

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

March 22, 2016

Outline

Introduction

- The problem of minimizing the maximum number of simultaneous open stacks arises in industrial environments:
  - A factory needs to supply specific combinations of products that have associated given demands;
  - A single machine manufactures all the products in batches and at each stage, a single product type is handled;
  - Whenever a customer places an order for a set of products, a new *open stack* is associated with it, meaning that physical space around the machine is assigned to it until the order is fulfilled - the *stack's closure*.

Introduction

- There is an implicit assumption of a physical limit on space around the machine because there is not enough free room around the machine to place all customers' orders simultaneously;
- Moving open stacks requires additional manpower and machinery, introducing risks to product's integrity;
- In order to better use that physical space, it is necessary to determine the sequence of the products' manufacturing.

Introduction

- The *Minimization of Open Stacks Problem* (MOSP) is defined on a binary sparse matrix M:
  - The n rows correspond to the customers's orders and the m columns correspond to each product type available;
  - Entry $m_{ij} = 1$ if customer i ordered product type j, and $m_{ij} = 0$ otherwise;
  - Matrix M holds the *consecutive ones property*[a].

[a] In each row all elements between two 1s are considered to have value 1, also called *fill-ins*. Those elements together define a *1-block*.

Formal Definition

- Given an $n \times m$ sparse binary matrix M, matrix M_π is the matrix obtained from a permutation π of the $\{1, 2, \ldots, m\}$ columns of matrix M, and M_π^1 is the matrix obtained from M_π that holds the consecutive ones property;
- The objective is to find a permutation π of columns of M such that the maximum sum of a column (including the fill-ins)[a] in M_π^1 is minimized.

[a] given by Z^π.

Example

Table 1 presents an example of sparse binary matrix M_π and M_π^1 for π=[5, 2, 4, 6, 3, 1]. $Z^\pi = 3$, on columns 2, 4, 6 and 3. Columns with maximum sum represent the bottleneck of problem and are called *critical columns*.

	1	2	3	4	5	6
1	1	1	0	0	0	0
2	1	0	1	0	0	0
3	0	0	0	1	1	0
4	0	0	0	1	0	1
5	0	1	0	0	1	0
6	0	0	1	0	0	1

	5	2	4	6	3	1
1	0	1	**1**	**1**	**1**	1
2	0	0	0	0	1	1
3	1	**1**	1	0	0	0
4	0	0	1	1	0	0
5	1	1	0	0	0	0
6	0	0	0	1	1	0

Table : M_π and M_π^1 for π=[5, 2, 4, 6, 3, 1]. Fill-ins are highlighted in bold font.

Motivation

- MOSP is a NP-hard problem;
- Its practical application;
- It models successfully a wide range of equivalent problems in a variety of contexts:
  - Interval Thickness;
  - Node Search Game;
  - Edge Search Game;
  - Narrowness;
  - Split Bandwidth;
  - Pathwidth;
  - Edge Separation;
  - Vertex Separation;
  - Modified Cutwidth;
  - Programmable Logic Array Folding;
  - Gate Matrix Layout.

Steepest Descent Approaches for the MOSP

Proposed Methods

- A graph search for generating an initial solution;
- A Variable Neighborhood Descent method;
- A simple Steepest Descent method.

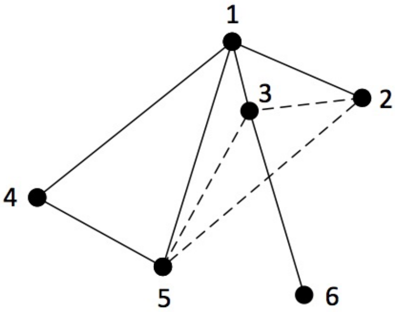
Current State of the Art

- Heuristic: Minimum Cost Node Heuristic (2004);
- Metaheuristic: Biased Random Key Genetic Algorithm (2014);
- Exact: A* (2009).

Graph Model

- Nodes represent customers;
- Edges represent products – no loops or parallel edges;
- Each customer induces a clique.

	1	2	3	4	5	6
1	1	0	0	1	1	0
2	1	1	0	0	1	0
3	0	0	1	0	0	1
4	1	1	1	0	0	0
5	0	1	1	0	1	0



Steepest Descent Approaches for the MOSP

Graph Search

- The main idea is to search for customers, rather than for products directly;
- Breadth-first search (BFS) in non-decreasing degree order.

Products Sequencing

- The BFS returns a list of customers;
- The products sequence is determined in a greedy fashion:
  - The list of customers is traversed, and once all customers that ordered a specific product are found, it is sequenced.

Steepest Descent Approaches for the MOSP

Variable Neighborhood Descent - VND

- Classic version – first time applied to the MOSP;
- Five neighborhood structures:
  - k-opt of grouped contiguous columns ($k = 2, 3, 4, 5$).
  - The best neighbor is improved by a insertion move (only bottleneck related columns).
- All possible movements explored partially
  - Randomly selected, within a limit (50%).
- All parameters tuned using the irace package.

Steepest Descent - SD

- Classic version;
- Local Search:
  - Insertion move – only bottleneck related columns;
  - Target position defined by similarity of nonzero elements.
- All possible movements explored partially
  - Randomly selected, within a limit (70%).
- All parameters tuned using the irace package.

Computational Environment

- Intel Core i7 3.6 GHz processor;
- 16 GB RAM;
- Ubuntu 14.04 LTS;
- Codes written in C++, compiled with gcc 4.8.4 and the -O3 optimization option.

Methods:

- Variable Neighborhood Descent (VND);
- Steepest Descent (SD);
- Minimum Cost Node heuristic (MCNh).

Instances

Six different data sets from the literature were considered, a total of 475 instances.

- SCOOP: 24 real world MOSP instances;
- Faggioli & Bentivoglio: 300 artificial MOSP instances;
- Challenge: 126 artificial MOSP instances;
- VLSI: 25 real world Gate Matrix Layout instances.

Steepest Descent Approaches for the MOSP

Average gap from Optimal Solutions (20 runs)

Method	SCOOP	Faggioli & Bentivoglio	Challenge	VLSI
VND	3.76%	2.04%	7.39%	3.93%
SD	2.15%	0.75%	0.6%	1.12%
MCNh	25.27%	13.72%	2.0%	12.92%

Steepest Descent Approaches for the MOSP

Average Running Times (20 runs)

Method	SCOOP	Faggioli & Bentivoglio	Challenge	VLSI
VND	12ms	19.46ms	215ms	64ms
SD	138ms	479ms	18s	4.08s
MCNh	<1ms	<1ms	<1ms	<1ms

Optimal Solutions Found

Method	SCOOP	Faggioli & Bentivoglio	Challenge	VLSI
VND	70.83%	81.33%	6.52%	76.00%
SD	87.50%	93.33%	86.96%	92.00%
MCNh	36.00%	37.00%	78.00%	72.00%

Conclusions

- This is the first time a VND application on MOSP is reported;
- The best heuristic in literature performed poorly on 3 problem sets – first time reported;
- VND is faster, but less accurate;
- Surprisingly, SD presented a very low gap and a good rate of optimal solutions found for most problem sets;
- Future work includes:
  - Variable Neighborhood Search;
  - Matheuristic – poor models currently.

Acknowledgments

This research has been supported by the National Council for Scientific and Technological Development (CNPq).

Questions?


рахмат
danke
謝謝
ngiyabonga
teşekkür ederim
Баярлалаа
спасибо
faafetai lava
mersi
kia ora
barka
welalin
tack
spas
dank je
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
vinaka
спасибі
blagodaram
thank you
gracias
kiitos
dankie
dhanyavad
nanni
nandri
mauruuru
koszonom
hvala
gracie
bayarlalaa
bedankt
enkosi
dziękuje
akun
dankon
aciu
хвала
asante
manana
obrigada
murakoze
tenki
djiere dieuf
tau
mochchakkeram
mamnun
дякую
chokrane
chnorakaloutioun
gratias ago
gracies
sulpay
taiku
go raibh maith agat
sobodi
dekuji
sagolun
sukriya
kop khun krap
mesi
didi madloba
ありがとう
tanemirt
grazie
arigatô
takk
dakujem
trugarez
najis tuke
kam sah hamnida
rahmat
terima kasih
rahmet
diolch
dhanyavadagalu
shukriya
merce
мерси
তোমাকে ধন্যবাদ
감사합니다
xiexie
ευχαριστώ
merci
obrigado