Norwegian University of Science and Technology Faculty of Information Technology, Mathematics and Electrical Engineering Department of Computer and Information Science

Master Thesis

Linear programming on Cell/BE

by

Åsmund Eldhuset

Supervisor: Dr.Ing. Lasse Natvig Co-supervisor: Dr. Anne C. Elster

Trondheim, June 1, 2009

Abstract

Acknowledgements

Contents

Co	onten	ts		vii
Li	st of	Figures		viii
Li	st of	Tables		ix
Li	sting	s		x
Li	st of	Symbo	ls and Abbreviations	xi
1	Intr	oductio	on	1
2	2.1	2.1.1 2.1.2 2.1.3 2.1.4	r programming	3 3 3 6 7 7 7
		2.2.2	Programming methods	7
3	Des	ign		9
4	Imp	lement	tation and testing	11
5	Eva	luation		13
6	Con	clusio	1	15
Bi	bliog	graphy		17
A	Sch	edule		21

List of Figures

List of Tables

Listings

List of Symbols and Abbreviations

Abbreviation	Description	Definition
LP	Linear programming	page 3

	1					
Chapter		_				

Introduction

Chapter				
Chapter	<i></i> .			

Background

(TODO)

Chapter introduction

Linear programming

(Natvig)

This section is primarily based on [?] and [?].

Do we need section introductions too?

problems that are

Problem formulation. Standard and slack forms

The term linear programming (LP) refers to a type of optimisation problems in which one seeks to maximise or minimise the value of a linear function of a set of variables¹. The values of the variables are constrained by a set of linear equations and/or inequalities. Linear programming is a fairly general problem type, and many important problems (TODO) can be cast as LP problems — for (other than those instance, network flow problems and shortest path problems (see [?]).

An example of a simple linear programming problem would be a factory that initially formulated makes two kinds of products based on two different raw materials. The profit as an LP problem) the company makes per unit of product A is \$10.00, and the profit of product B is \$12.50. Producing one unit of A requires 2 units of raw material R and 3 units of raw material S; one unit of B requires 3 units of R and 1.5 units of S. The company possesses 100 units of raw material R and 50 units of raw material S. We make the simplifying assumptions that all prices are constant and cannot be affected by the company, and that the company is capable of selling everything it produces. The company's goal is to maximise the profit, which can be described as $10.00x_1 + 12.50x_2$ where x_1 is the number of units of product A and x_2 is the number of units of product B. The following constraints are in effect:

¹Hence, LP is not (as the name would seem to suggest) a programming technique.

- $2x_1 + 3x_2 \le 100$ (the production of A and B cannot consume more units of raw material R than the company possesses)
- $3x_1 + 1.5x_2 \le 50$ (same for raw material S)
- $x_1, x_2 \ge 0$ (the company cannot produce negative amounts of its products)

We will use this example throughout this section.

Note that in regular LP problems, one cannot restrict the variables to be integers — in fact, adding this requirement makes the problem NP-hard². It is also, in general, a requirement that all variables are nonnegative. Real-world problems involving variables that may be negative as well as positive can still be modeled by writing each original variable as a difference of two nonnegative variables.

	\overline{z}	x_1	x_2	x_3
z				
x_4				
x_5				

Throughout this report, we will consistently use n to refer to the number of variables and m to refer to the number of inequalities. The variables will typically be (TODO: spell "label(l)ed") x_1 through x_n .

The function to be optimised is called the *objective function*. In the real world situation that gives rise to an optimisation problem, the function may contain a constant term, but it can be removed since that will affect all possible solutions in the same way. The objective function can then be written as $f = c_1x_1 + c_2x_2 + c_3x_1 + c_3x_2 + c_3x_3 + c_3x_3$ $\ldots + c_n x_n = \sum_{j=1}^n c_j x_j$, where the c_j are called the *coefficients*. (TODO)

Vonnegativity of ariables, which is world prolems.

han allowed? consistency he standard/slack orms? How to indent?

often the case in real Standard form The equations and inequalities that (together with the objective function) constitute an LP problem may be represented in different forms. We shall first consider the standard form, in which only less-than-or-equal-to in-Why are not less- equalities with all variables on the left hand side are allowed. (TODO) A problem containing equalities of the form $a_1x_1 + \ldots + a_nx_n = b$ (Natvig) may be Should I label the co-rewritten by splitting each equality into two inequalities: $a_1x_1 + \ldots + a_nx_n \leq b$ efficients a_{i1}, \ldots, a_{in} and $-a_1x_1 - \ldots - a_nx_n \leq -b$. Also, the goal must be to maximise the objective nstead to maintain function — if the original problem is to minimize f, we let our objective function with be -f. A linear program in standard form can be expressed as follows: (TODO)

²NP-hardness is a term from complexity theory, which deals with the relative difficulties of solving different kinds of problems. The only known algorithms for solving NP-hard problems require an amount of time that is exponential in the size of the problem, which renders those algorithms useless for many real life problem sizes.

5

Maximise

$$f = \sum_{j=1}^{n} c_j x_j$$

with respect to

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \text{ for } i = 1, \dots, m.$$
$$x_1, \dots, x_n \le 0$$

Slack form The other common representation is *slack form*, which only allows a set of equations (and a nonnegativity constraint for each variable). An inequality of the form $a_1x_1 + \ldots + a_nx_n \leq b$ is converted to an equation by adding a *slack variable* w. Together with the condition that $w \geq 0$, the equation $a_1x_1 + \ldots + a_nx_n + w = b$ is equivalent to the original inequality (whose difference, or "slack", between the left and right hand sides is represented by w). A linear program in slack form can be expressed as follows:

Maximise

$$f = \sum_{j=1}^{n} c_j x_j$$

with respect to

$$w_i = b_i - \sum_{j=1}^n a_{ij} x_j, \text{ for } i = 1, \dots, m.$$
 $x_1, \dots, x_n \le 0$

A proposed solution of a linear program (that is, a specification of a value for each variable) is called:

Feasible if it does not violate any of the constraints

Infeasible if it violates any constraint

Basic (TODO)

Optimal if it is feasible and no other feasible solutions yield a higher value for the objective function

(TODO)

The ??? theorem ((TODO: citation)) states that the optimal solution of a linear program, if it exists, occurs when m variables are set to zero and the m others are nonzero

Simplex algorithm 2.1.2

The simplex algorithm, developed by (TODO)[?], was the (TODO) systematic algorithm developed for solving linear programs. It requires the program to be in slack form. (TODO) The nonnegativity constraints are not represented explicitly anywhere. (TODO)

Tableaux The variables in the leftmost column are referred to as the basic variables, and the variables inside the tableau are called nonbasic variables. At any ???? Dantzig stage of the algorithm, the set of the indices of the basic variables is denoted first? \mathcal{B} , and the set of nonbasic indices is denoted \mathcal{N} . It should be noted that the slack form must have been created from a standard form, because this ensures that there are m slack variables, where each slack variable occurs in excactly one equation. Initially, the set of basic variables is the set of slack variables, and the sizes of the basic and nonbasic sets are constant, with $|\mathcal{B}| = m$ and $|\mathcal{N}| = n$.

For now, let us assume that the solution that is obtained by setting all nonbasic variables to zero is feasible (which is the case only if none of the b_i are negative); we will remove this restriction later. This trivial solution will provide a lower bound for the value of the objective function (namely, the constant term). We will now select one nonbasic variable x_i and consider what happens if we increase its value (since all nonbasic variables are currently zero, we cannot decrease any of them). Since our goal is to maximise the objective function, we should select a variable whose coefficient c_i in the objective function is positive. If no such variables exist, we cannot increase the objective function value further, and the current solution (the one obtained by setting all nonbasic variables to zero, so that $f = c_0$) is optimal — we can be certain of this since linear functions do not have local maxima.

It seems reasonable to select the variable with the greatest coefficient, say, x_l . How far can we increase this variable? Recall that each line in the tableau expresses one basic variable as a function of all the nonbasic variables; hence we can increase x_l until one of the basic variables becomes zero. Let us look at row i. If a_{ij} is negative, the value of w_i will decrease as x_l increases, so the largest allowable increase is limited by the current value of w_i — which is b_i , since all nonbasic variables were set to zero. Thus, by setting $x_l = -\frac{b_i}{a_{ij}}$, w_i becomes zero. However, other equations may impose stricter conditions. By looking at all rows where a_{ij} is negative, we can determine $\min \left(-\frac{b_i}{a_{il}}\right)$ and set x_j equal to it. If all a_{il} are nonnegative, we can increase x_l indefinitely without any w_i ever becoming negative, and in that case, we have determined the program to be *unbounded*; the algorithm should report this to the user and terminate.

The next step, called *pivoting*, is an operation that exchanges a nonbasic variable and a basic variable. The purpose of pivoting is to produce a new situation in which no b_i is negative, so that we can repeat the previous steps all over again. The nonbasic variable that was selected to be increased, x_i , is called the *entering* variable, since it is about to enter the collection of basic variables. The leaving variable to be removed from said collection. (TODO) We can eliminate the en- how to find it? tering variable from (and introduce the leaving variable into) the set of nonbasic variables (the "main" part of the tableau) by rewriting the selected equation and adding appropriate multiples of it to each of the other equations:

Initialisation

The algorithm presented so far is capable of solving linear programs whose initial basic solution (the one obtained by setting all nonbasic variables to 0) is feasible. (TODO) This may not always be the case. We get around this by intro- Phase I and Phase II ducing an auxiliary problem which based on the initial problem that is guaranteed

Degeneracy

Example

We will now demonstrate one iteration of the simplex algorithm, on the following problem: (TODO)

2.1.3 Interior point algorithms

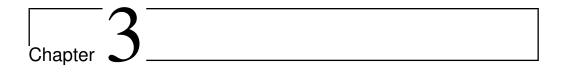
Use of LP to solve advanced flow problems

(TODO)

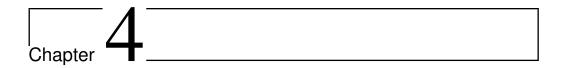
Consult Miriam or this

Cell Broadband Engine 2.2

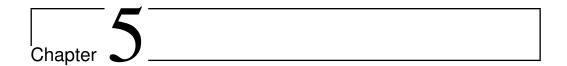
- Architecture 2.2.1
- 2.2.2 Programming methods



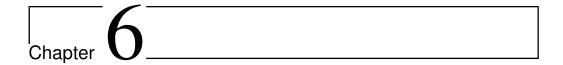
Design



Implementation and testing



Evaluation



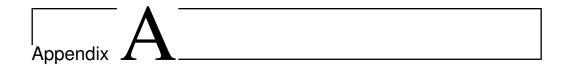
Conclusion

(TODO)

Future work

Bibliography

Appendices



Schedule

This appendix will obviously be deleted before submission.

Week 8 Finished the implementation of a dense Simplex for a regular CPU and test with netlib datasets. Implement a vectorised (SIMD) dense Simplex on the PPE

Week 9 Struggle with numerical instability

Week 10 Implement a vectorised dense Simplex running in parallel on the SPEs

Week 11 Study interior point algorithms

Week 12 Implement a dense, non-parallelised interior point algorithm

Week 13 Decide on whether to pursue simplex or interior point. Making a test plan. Experiment with different approaches to sparse storage; look into numerical stability with single-precision values

Week 14 First draft of report

Week 15 Easter vacation

Week 16 Same as week 13

Week 17 Look into autotuning?

Week 18

Week 19

Week 20 Performance measurements and graphing

Week 21 Frenetic report writing

Week 22 — " —

Week 23 Ordinary submission deadline. Will try to submit as close to this date as possible

Week 24

Week 25

Week 26

Week 27 Natvig goes on vacation

Week 28

Week 29 Final deadline: July 19