620-361 Operations Research Techniques and Algorithms

Penalty methods for n-d optimisation
A survey

Student 1: Y.Chan Y.Chan@ms.unimelb.edu.au
Student 2: C.Burt C.Burt@ms.unimelb.edu.au

Department of Mathematics and Statistics
University of Melbourne

1 Introduction

Penalty methods are designed to solve n-d constrained optimisation problems by instead solving a sequence of specially constructed unconstrained optimisation problems [1]. This is achieved by adding a penalty to the objective function for all points that do not satisfy the constraints. Associated with the severity of the penalty is a parameter, c, which is related to the accuracy of the approximation of the constrained problem ny the unconstrained problems. In general, as $c \to \infty$ the approximation becomes more accurate, although some penalty functions yield exact solutions [2].

We are interested in understanding the applications for which these methods can be suitably and successfully applied. In particular, we are interested in studying the performance of these algorithms to solve the following two problems:

P1:

$$\min x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$$

subject to

$$\begin{aligned} -x_1^2 - x_2^2 - x_3^2 - x_4^2 - x_1 + x_2 - x_3 + x_4 + &\geq 0 \\ -x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2 + x_1 + x_4 + 10 &\geq 0 \\ -2x_1^2 - x_2^2 - x_3^2 - 2x_1 + x_2 + x_4 + 5 &\geq 0 \end{aligned}$$

and

P2:

$$\min[1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$$

subject to

$$\mathbf{x}^T\mathbf{x} = 1$$

By studying the behaviour of penalty methods to these two problems, we hope to gain insight into the methods that will enable us to choose the most suitable method in future applications to different problems.

2 Objectives

The aim of this project is to gain a deep understanding of the application of penalty methods to n-d constrained optimisation problems. In particular, we will:

1. perform a literature review of penalty methods for n-d constrained optimisation;

2. determine which of the Penalty method algorithms performs “best” on the two problems outlined in the Introduction [Section 1], using suitable performance criteria and measures;

3. analyse both the algorithms and problems (P1 and P2) so that we may provide a discussion as to why particular algorithms outperform others for these cases;

4. extend this discussion to general cases in a bid to understand when the penalty methods might be useful; and

5. consider real life situations and applications which might be well solved by these methods.

Achieving these objectives will put us in a good position to appropriately apply penalty methods in applications not considered in this course.

3 Research method

We study the available literature for methods that fall into the following categories: quadratic penalty, logarithmic barrier, exact penalty functions, and augmented Lagrangian. We are primarily interested in the general techniques rather than specialised algorithms designed for rare cases. We will implement these algorithms in Matlab and adopt P1 and P2 as test cases for the comparative analysis.

We now establish a basis for comparison of these methods. With penalty methods, we are very concerned with how well the unconstrained problem (i.e., the penalty function) approximates the constrained problem. In particular we are interested in the trade-off between **convergence rate** and **approximation accuracy** as $c \to \infty$. We are also interested in other implementation issues with the algorithms, such as **ease of implementation**, **number of iterations** required, **information** required, **robustness**, and **sensitivity to initial conditions**.

We measure the convergence rate in two ways - we examine the relevant theorems of convergence for each algorithm in our study and provide a proof of the theorem. This gives us an indication of the theoretical rate of convergence. We then track both the iterations and the improvement in the estimates over these iterations, which will act to validate the theoretical convergence rate.

We measure the ease of implementation on a simple scale of easy (one simple routine), medium (a handfull of simple routines and functions) and hard (several functions and routines with at least one requiring great thought). We monitor the information required for each algorithm in terms of function calculations, derivative calculations, matrix calculations (including inverses), and checks for satisfiability of conditions (such as positive definiteness).

The robustness and sensitivity to initial conditions are related criteria. We choose to assess the robustness of the algorithm by creating deliberate errors in the model (namely the penalty function), and observe how well the algorithm performs (with respect to approximation accuracy and iterations required). We analyse sensitivity to initial conditions the obvious way - by changing the initial conditions and observing the change in the behaviour of the algorithms (with respect to iterations required).

We can better understand why these algorithms might perform a certain way on the test cases by analysing the test cases themselves. We will study these problems in terms of positive definiteness of the coefficient matrix, and any other problem related issues that might come to light during the literature survey.

4 Timeline

This is a basic project management task to help the group realise deadlines and allocate tasks.

Week	1	2	3	4	5	6	7	8	9	10	11	12
Literature survey	•	•	•	•								
Implement algorithms		•	•	•	•							
Analyse problems						•	•	•	•			
Write report									•	•	•	•
Prepare oral presentation											•	•

Table 1: Our timeline for this semester.

References

[1] Freund, R. (2004) *Penalty and barrier methods for constrained optimization.* Available online: `http://ocw.mit.edu/NR/rdonlyres/Sloan-School-of-Management/15-084JSpring2004/A8E10BC8-6B04-4D64-94F2-FB697408B1FF/0/lec10_penalty_mt.pdf`

[2] Luenberger, D. (2003) *Linear and nonlinear programming.* Available online: `http://books.google.com.au/`

[3] Pillo, G. (1994) Exact penalty methods. In *E. Spedicato (ed), Algorithms for continuous optimization*, pp 209-253. Springer. Availble online: `http://books.google.com.au`