POLYNOMIAL TIME HEURISTIC OPTIMIZATION METHODS APPLIED TO PROBLEMS IN COMPUTATIONAL FINANCE

PH.D. DISSERTATION OF

Fogarasi Norbert, M.Sc.

Supervisor:
Dr. Levendovszky János, D. Sc.
Doctor of the Hungarian Academy of Sciences

Department of Telecommunications
Budapest University of Technology and Economics

Budapest, 2014

„I was like a boy playing on the sea-shore, and diverting myself now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me." (Isaac Newton)

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Janos Levendovszky for his ongoing guidance and direction for this research work.

I would also like to thank Kalman Tornai, Robert Sipos, Farhad Kia, dr. Gabor Jeney and dr. Antal Ivanyi for useful discussions and collaboration on some of the problems presented in the theses.

I would like to thank Morgan Stanley, my employer, for allowing me to take a leave of absence to focus on this research and for providing data to prove the viability of the methods in real-world applications.

Finally, and most importantly, I would like to thank my dear wife, Ildikó and my children, Ágnes and Sámuel for their endless patience and support for completing this work.

Executive Summary

Computational finance is one of the most dynamically advancing fields of research over the last 60 years [80]. With the advancement in computational hardware speed and heuristic techniques, many problems which were once believed to be intractable have been solved or sufficiently good approximations have been found. Recent research works in computational finance focus on finding fast approximations which can be applied in real-time, high frequency algorithmic trading applications ([12],[26],[56],[80]) as well as finding reliable approximate solutions to hard problems which have desirable runtime and generalization characteristics ([11],[22],[23],[28],[57]).

This dissertation has the same aims in two critically important research areas: portfolio selection and optimal scheduling, both of which have profound importance in present day financial computing. Optimal portfolio selection is one of the very first problems posed in the field in the 1950's and remains relevant to this day. Rather than the classical mean-variance approach [62], we look at the problem of selecting sparse portfolios which maximize mean reversion. I apply various search heuristics and stochastic search techniques which find approximate solutions to the resulting NP hard problem in polynomial time. My related results can be found in [1],[2],[5]. Scheduling theory is also a very pertinent area of research with applications in many different disciplines, including computational finance. I look at a classical problem in the field and develop a number of near-optimal heuristics to find good approximate solutions with extremely favorable scalability properties. My related results can be found in [3],[4].

In both cases, I show the viability and superiority of my methods on both a vast number of randomly generated synthetic data series as well as on real historical data, such as historical time series of major equity indices (S&P 500) and interest rate swap rates. For selecting sparse portfolios which maximize mean reversion, I do not only explain how to elegantly adapt stochastic search techniques to the problem, but I make a number of improvements upon the results found in the literature in terms of parameter fitting based on historical data for both the VAR(1) and the Ohrnstein-Uhlenbeck models. For scheduling tasks on identical machines such that total weighted tardiness is minimized, I show how the problem can be converted to a quadratic optimization form to be able to apply well-known heuristic algorithms. I further improve the method by smart choice of initial point for the iterative algorithm and suggest a method to further improve by perturbing this smart initial value. Using these results, large-scale Monte Carlo simulation computations can be significantly sped up for financial computations.

In both cases, the suggested methods result in double digit percentage improvement over the next best known heuristic method on both generated and real historical time series. I also conclude that the newly developed heuristic methods have good generalization properties which allow them to

be applied successfully to a broad range of problems within computational finance. I also suggest further ideas and directions of research for the specific problems and suggested heuristic methods.

These results contribute significantly to making financial computing faster and more reliable for financial institutions, which has an impact on the efficiency of the banking system and thus benefit society as a whole.

POLYNOMIAL TIME HEURISTIC OPTIMIZATION METHODS APPLIED TO PROBLEMS IN COMPUTATIONAL FINANCE

List of Figures

Fig. 12: CPU runtime (in seconds) versus total number of assets *n,* to compute a full set of sparse portfolios, with cardinality ranging from 1 to *n,* using the different algorithms.

Fig. 13: Typical example of convergence trading over 250 time steps on a sparse mean-reverting portfolio. Weighted mix of L=5 assets were selected from n=10-dimensional generated VAR(1) process by simulated annealing during the first 20 time steps. A profit of $1,440 was achieved with an initial cash of $100 after 85 transactions.

Fig. 14: Histogram of profits achieved over 2,000 different generated VAR(1) series by simulated annealing.

Fig. 15: Comparison of minimum, maximum, average and final return on S&P500 historical data of the greedy and simulated annealing methods for sparse mean-reverting portfolios of size L=3 and L=4

Fig. 16: Comparison of minimum, maximum, average and final return on historical US swap data of the truncation method for sparse mean-reverting portfolios of cardinality L=1-8

Fig. 17: Summary flow diagram of the heuristic approach to solving the Total Weighted Tardiness scheduling problem.

Fig. 18: Overall flow diagram of my novel HNN approach to solving scheduling problems

Fig. 19: Block diagram of the SHNN method

Fig. 20: Block diagram of the PSHNN method

Fig. 21: Illustration of the advantage of multiple perturbed starting points in finding the global minimum

Fig. 22: Block diagram of the PLWPF method

List of Tables

List of Algorithms

CHAPTER 1

INTRODUCTION

Computational finance is a branch of applied computer science that deals with problems of practical interest in finance [80]. It is a relatively new discipline the birth of which can be traced back to the work of Harry Markowitz in the 1950s and since then has provided many problems of deep algorithmic interest. Presently, it encompasses various new numerical methods in the fields of optimization, statistics, and stochastic processes developed for different financial applications.

Markowitz conceived of the portfolio selection problem as an exercise in mean-variance optimization. His key finding was that diversification, as a means of reducing the variance and therefore increasing the predictability of an investment portfolio is of critical importance, even at the cost of reducing expected return [62]. For his groundbreaking work, Markowitz received the Nobel Prize in 1990. There are many other problems in finance which have been raised since the 1950's such as algorithmic trading, the validation of efficient markets hypothesis, option pricing, financial time series analysis and prediction.

The area of algorithmic trading, the use of electronic platforms for entering trading orders with an algorithm deciding on aspects of the order such as timing, price or quantity, has attracted a lot of attention over the last decade. In particular, high frequency trading has gained a lot of ground in international markets, in which computers make algorithmic decisions on high frequency, often sub-second, tick-by-tick data, before human traders are capable of processing the information they observe. It is estimated that as of 2009, high frequency trading algorithms are responsible for over 70% of all equity trading volume in the US [39], so the area has also attracted a lot of academic attention in recent years. At the same time, parallel computational techniques on hardware, such as GPGPU and FPGA have also evolved, so traditional methods which could historically only been used in low or medium frequency trading can now be applied in real-time settings. On the other hand, many of the problems which arise during the development of optimal trading algorithms are of high complexity which makes the application of fast approximation methods necessary for practical use in high frequency trading.

Another developing area of computational finance is the increased use of Monte Carlo techniques for simulation of market processes under certain stochastic model assumptions. This large scale task requires the use of an increasing amount of computational resources and gives rise to problems of optimal usage of the resources via scheduling. Many of the combinatorial problems of optimal scheduling have been proven to be of exponential or NP hard complexity which have made the use of polynomial approximation techniques necessary.

The main motivation for research into these topics is their direct applicability to the real world. Many financial services companies, including banks, hedge funds and portfolio managers work on assembling portfolios with predictable characteristics which can be used as a reliable form of investment for individuals and institutions. The approach to maximize predictability by considering mean reverting portfolios is a relatively new idea, introduced in [26] which does not follow the Markowitz approach. The practical application is the implementation of so-called *convergence trading* algorithms which take advantage of the predictable nature of the portfolio. The sparseness constraint, which makes the problem computationally difficult, is very important in real life, as it ensures that the transaction cost required to purchase the portfolio is limited. My contribution is the application of stochastic search techniques to this NP hard problem which find good approximate solutions within polynomial time.

However, even if an algorithm with theoretically desirable runtime characteristics has been developed, such as above, the practical viability of the methodology often depends on technical details of the implementation. In practice, the computations can often be broken up into independent tasks which then need to be scheduled on a limited number of resources. This technical problem is, in itself, one that has attracted considerable amount of attention in the field. Given its importance to the overall applicability of heuristic methods to problems in finance, I decided to focus on it as the second main problem of my dissertation. Finding schedules for tasks running on identical machines, which minimize the total weighted tardiness has been proven to be NP hard, but I have been able to develop heuristic algorithms which run in polynomial time and have better scalability characteristics than other methods found in the literature.

The below table summarizes the practical application of the problems examined in my dissertation.

Theoretical problem description	**Practical applications in computational finance**
Identifying sparse, mean reverting portfolios	• Investment portfolio selection for banks, hedge funds and portfolio managers • Low frequency electronic trading strategy based on convergence trading • High frequency, real-time electronic trading strategy based on convergence trading
Finding schedules for tasks running on identical machines which minimize total weighted tardiness	• Real-time scheduling of Monte Carlo simulations for intraday pricing • End-of-day scheduling of Monte Carlo simulations for daily mark to market

	• Overnight scheduling of Monte Carlo simulations for risk and sensitivity calculations

Table 1. Practical applications in computational finance of the problems examined in the dissertation

1.1. Objectives of the dissertation

One of the basic aims of computational finance over the last 50 years has been the development of efficient algorithms which leverage the availability of hardware which has become faster and faster over the years. In today's world, the practical running speed of the algorithm is an essential feature which determines its scope of applicability. If the running speed can be improved to be sub-second then the algorithm can be applied in high frequency algorithmic trading, but if the algorithm takes longer to run, it can only be applied in low frequency trading applications. High frequency trading has become crucially important in today's financial world where new quote and trading information arrives tick by tick in sub-second intervals, so the more quickly an automated algorithm can react, the more likely that it will be able to act upon up-to-date information.

At the same time, many of the fundamental problems which have arisen in the field have been shown to be of exponential complexity or NP hard. As a result, the fundamental objective of computational finance has been to find numerical algorithms yielding approximate solutions within practically viable timeframes to these problems.

The present dissertation has the same aim for two specific problems which are of crucial importance to financial institutions today:

1) the problem of finding sparse portfolios, based on historical data, which maximize predictability (or mean reversion); and
2) the problem of finding optimal schedules for a set of identical computational resources for jobs (Monte Carlo simulations) with prescribed sizes, weights and deadlines which minimize total weighted tardiness.

Even though these two problems are seemingly rooted in different areas they are brought together in the area of computational finance (portfolio optimization and optimal resource scheduling), and the theme of finding fast, heuristic approximate solutions to NP hard problems with applications in finance, is common. As a result, the theses provide novel and important contributions to the field of computational finance which have very practical applications. Thus the objective of the dissertation can be stated as aiming to bridge the gap between NP hard problems and real-time or near real-time solutions which can be used in high frequency trading.

1.2. Models and methods used in the research

In order to achieve the results presented in the dissertation, a number of models and computational methods have been used and developed. These are outlined in this section.

For solving the problem of sparse, optimally mean reverting portfolios, I follow the construction outlined in [26]. d'Aspremont made a number of assumptions in the construction: he assumed the underlying processes follow a VAR(1) model and that the resulting mean reverting portfolios can be described by the Ornstein-Uhlenbeck equation [70]. For the estimation of model parameters, I used the Moore-Penrose pseudoinverse [68] to solve the equation arising during the maximum likelihood estimation of the recursion matrix and I developed a novel recursive method for estimating the covariance matrix. Having estimated the parameters of the VAR(1) model, I mapped the prediction maximization problem to a generalized eigenvalue problem with cardinality constraint which has been shown to be NP hard [67]. Therefore, I applied simulated annealing [48] to the problem, in order to find a polynomial time approximation and I compared this to a number of benchmark methods. These include exhaustive search [1], greedy search [26] and a novel heuristic which I developed based on truncation. Having found the portfolio, I developed a novel method to find its long-term mean using pattern matching. Finally, the economic viability of the method was verified by running numerical simulations based on a novel convergence trading strategy which I developed based on a decision theoretic formulation.

For solving the problem of finding optimal schedules on identical machines which minimize the total weighted tardiness of jobs, I used a binary scheduling matrix model to represent the schedules [3]. Furthermore, I studied a number of existing heuristic methods (EDD, WSPT, LWPF, LBS) which I subsequently used as benchmarks for evaluating the performance of my novel approach. I used matrix algebraic transformations to convert the problem to quadratic form and built the constraints into the objective function, using heuristic constants. This allowed me to use Lyapunov convergence via the Hopfield neural network [42] to find polynomial time approximate solutions to this NP hard problem. I further improved upon this method by introducing random perturbation to the intelligently selected initial point [4].

All model and method implementations were performed in the MATLAB computational environment, where numerical simulations were run on both synthetic and real, historical data for both problems at hand. The results presented in the dissertation are based on the numerical simulations performed in this way for both problems.

Problem investigated	Sparse, mean reverting portfolio selection	Task scheduling on identical machines
Models used	VAR(1) model Ornstein-Uhlenbeck model	Binary scheduling matrix
Methods applied	• Maximum likelihood estimation • Moore-Penrose pseudoinverse • Novel recursive covariance matrix estimation • Benchmark methods (Exhaustive search, greedy search, novel truncation method) • Simulated annealing • Novel long-term mean estimation based on pattern matching • Novel convergence trading based on decision theoretic formulation	• Heuristic optimization (EDD, WSPT, LWPF, LBS) benchmark methods • Combinatorial optimization of quadratic forms • Lyapunov convergence • Hopfield neural network • Random perturbations
Sections in Dissertation	• Chapter 2 (pp.21-49)	• Chapter 3 (pp. 50-72)
Validation	Numerical simulations on synthetic and real historical data in MATLAB	

Table 2. Models, methods and validation used to derive the results of this dissertation.

1.3. Structure of the dissertation

In chapter 2, I define and study the problem of finding sparse portfolios which maximize mean reversion. After formally defining this NP hard problem I present existing heuristics and also present a number of heuristics which I have developed, including truncation, exhaustive and simulated annealing methods. I then go on to explain classical ways of fitting the model and make a number of new contributions to this area. Finally, I present the results of these new methods on both synthetically generated data and real historical financial time series.

In chapter 3, I define and study the problem of finding schedules for tasks on identical machines which minimize the total weighted tardiness. After formally defining this NP hard problem, I present existing heuristics and also a number of new heuristics which I have developed, including the Hopfield Neural Network (HNN) approach, the Smart Hopfield Neural Network (SHNN) approach and the Perturbed Smart Hopfield Neural Network (PSHNN) approach.

In chapter 4, I summarize the main results of the dissertation, draw some general conclusions regarding the methodology used, and present directions for future research.

Finally, in the appendix I formally define my notation and abbreviations as well as presenting a list of my own publications and a full bibliography for this work.

CHAPTER 2

OPTIMAL SPARSE, MEAN REVERTING PORTFOLIO SELECTION

In this chapter, I present discussion of the problem of finding sparse portfolios which maximize mean reversion. This is a very recently posed problem in [26] which differs greatly from the classical mean-variance approach to portfolio selection [62]. The intuition is that by selecting predictable portfolios, one can develop profitable, so called "*convergence trading*" strategies which can outperform the classical buy and hold strategy associated with minimizing risk for a given level of portfolio return. By introducing the sparseness constraint, we ensure that simple, easily interpretable portfolios are selected and that transaction costs are kept small.

2.1. Introduction

Ever since publication of the seminal paper of Markowitz [62], the selection of portfolios which are optimal in terms of risk-adjusted returns has been an active area of research both by academics and financial practitioners. A key finding of Markowitz is that diversification, as a means of reducing the variance and therefore increasing the predictability of an investment portfolio, is of paramount importance even at the cost of reducing expected return. The resulting, so-called "mean-variance" approach to portfolio selection involves tracing out a surface of efficient portfolios which can be found via quadratic programming and has been applied successfully by practitioners ([49] [35]). Recent improvements to the model have considered including more complex measures of risk [32], transaction costs ([10],[85]) and sparsity constraints ([82], [12]) to this basic model. These additional constraints have made the portfolio selection problem significantly more complex, justifying the use of various heuristic methods, such as genetic algorithms ([57],[15]), neural networks [36] and even simulated annealing ([23],[13]). A comprehensive study of the metaheuristics in portfolio selection can be found in [28].

At the same time, mean reversion, as a classic indicator of predictability, has also received a great deal of attention over the last few decades. It has been shown that equity excess returns over long horizons are mean-reverting and therefore contain an element of predictability ([73], [33],[61]).

While there exist simple and reliable methods to identify mean reversion in univariate time series, selecting portfolios from multivariate data which exhibit this property is a much more difficult problem. This can be approached by the Box-Tiao procedure [21] to extract cointegrated vectors by solving a generalized eigenvalue problem.

In his recently published article, d'Aspremont posed the problem of finding sparse portfolios which maximize mean reversion [26]. The impetus is that by finding mean reverting portfolios, one can develop conversion trading strategies to produce profits, as opposed to the buy and hold strategy associated with portfolios which maximize excess returns or minimize risk. Sparseness, he argues, is desirable for reducing transaction costs associated with convergence trading as well as for increasing the interpretability of the resulting portfolio. He developed a new approach to solve the problem by using semidefinite relaxation and compared the efficiency of this solution to the simple greedy algorithm in a number of markets.

My contribution to this topic is manyfold. First, I developed a new method for estimating parameters of the VAR(1) process, which also provides a model fit measure. Furthermore, I introduce a novel method for estimating the long-term mean of an Ornstein-Uhlenbeck process based on pattern matching. I also develop a number of simple heuristic methods and a new method for portfolio selection, adapting the simulated annealing method and finally develop a simple convergence trading algorithm through which I show the practical viability of my methods. Fig. 1 illustrates the end-to-end framework I have developed for finding heuristic solutions to this NP hard problem.

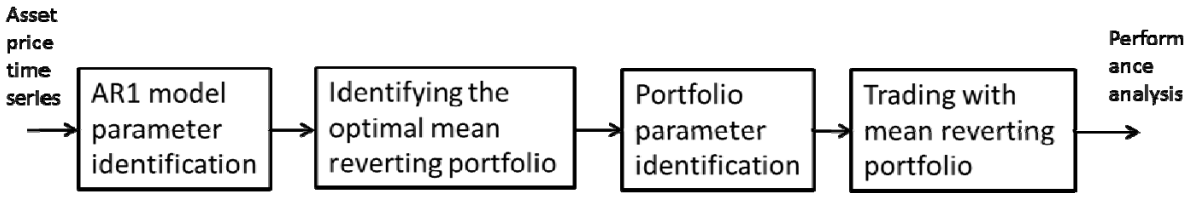

Fig. 1: Overall framework for identifying and trading sparse mean reverting portfolios

2.2. Modeling mean reverting portfolios

The problem I examine is the selection of sparse, mean reverting portfolios which follow the so-called Ornstein-Uhlenbeck process [70]. Let $s_{i,t}$ denote the price of asset *i* at time instant *t*, where $i = 1,...,n$ and $t = 1,...,m$ are positive integers. I form portfolios, of value p_t of these assets with coefficients x_i, and assume they follow an Ornstein-Uhlenbeck process given by:

$$dp_t = \lambda(\mu - p_t)dt + \sigma dW_t, \tag{1}$$

where W_t is a Wiener process and $\lambda > 0$ (mean reversion coefficient), μ (long-term mean), and $\sigma > 0$ (portfolio volatility) are constants.

Considering the continuous version of this process and using the Ito-Doeblin formula [44] to solve it, I get:

$$p(t) = p(0)e^{-\lambda t} + \mu\left(1 - e^{-\lambda t}\right) + \int_0^t \sigma e^{-\lambda(t-s)} dW(s) \tag{2}$$

which implies that

$$E[p(t)] = p(0)e^{-\lambda t} + \mu\left(1 - e^{-\lambda t}\right) \tag{3}$$

and asymptotically

$$\lim_{t\to\infty} p(t) \sim N\left(\mu, \sqrt{\frac{\sigma^2}{2\lambda}}\right) \tag{4}$$

For trading, λ is a key parameter [26], as it determines how fast the process returns to the mean, as well as inversely indicating the level of uncertainty around the mean (via the standard deviation of the asymptotic Gaussian distribution). Hence, the larger the λ, the more suitable the mean reverting portfolio for convergence trading, as it quickly returns to the mean and contains a minimum amount of uncertainty around the mean. Therefore, I will be concerned with finding sparse portfolios which are optimal in the sense that they maximize λ.

2.3. Mean reverting portfolio as a generalized eigenvalue problem

In this section, I view asset prices as a first order, vector autoregressive VAR(1) process. This is a very commonly used statistical model in financial time series analysis which generalizes the univariate autoregressive (AR) models by allowing for more than one evolving variable. VAR modeling is preferred in most related works ([17],[25],[26],[27]), as it does not require too many assumptions about the forces controlling the evolution of variables as do structural models with simultaneous equations. The model can be applied in most econometric settings where it is reasonable to hypothesize that variables may affect each other intertemporally [81].

Let $s_{i,t}$ denote the price of asset *i* at time instant *t*, where $i = 1,...,n$ and $t = 1,...,m$ are positive integers, and assume that $\mathbf{s}_t^T = (s_{1,t},...,s_{n,t})$ is subject to a first order vector autoregressive process, VAR(1), defined as follows:

$$\mathbf{s}_t = \mathbf{A}\mathbf{s}_{t-1} + \mathbf{W}_t, \tag{5}$$

where **A** is an $n \times n$ matrix and $\mathbf{W}_t \sim N(0, \sigma I)$ are i.i.d. noise terms for some $\sigma > 0$.

One can introduce a portfolio vector $\mathbf{x}^T = (x_1, ..., x_n)$, where component x_i denotes the amount of asset *i* held. In practice, assets are traded in discrete units, so $x_i \in \{0, 1, 2, ...\}$ but for the purposes of this analysis I allow x_i to be any real number, including negative ones, which denote the ability to short sell assets. Multiplying both sides by vector **x** (in the inner product sense), I obtain

$$\mathbf{x}^T \mathbf{s}_t = \mathbf{x}^T \mathbf{s}_{t-1} \mathbf{A} + \mathbf{x}^T \mathbf{W}_t \tag{6}$$

Following the treatment in [26] and [21], I define the *predictability* of the portfolio as

$$\nu(\mathbf{x}) := \frac{\mathrm{var}(\mathbf{x}^T \mathbf{A}\mathbf{s}_{t-1})}{\mathrm{var}(\mathbf{x}^T \mathbf{s}_t)} = \frac{E(\mathbf{x}^T \mathbf{A}\mathbf{s}_{t-1}\mathbf{s}_{t-1}^T \mathbf{A}^T \mathbf{x})}{E\left(\mathbf{x}^T \mathbf{s}_t \mathbf{s}_t^T \mathbf{x}\right)}, \tag{7}$$

provided that $E(\mathbf{s_t}) = 0$, so the asset prices are normalized for each time step. The intuition behind this portfolio predictability is that the greater this ratio, the more $\mathbf{s_{t-1}}$ dominates the noise, and therefore the more predictable $\mathbf{s_t}$ becomes. Therefore, following the construction in [26], I will use this measure as a proxy for the portfolio's mean reversion parameter λ in (1). The intuition behind this substitution is that the predictability measure is equivalent to the mean reversion parameter in the stationary case. Maximizing this expression will yield the following optimization problem for finding the best portfolio vector $\mathbf{x_{opt}}$:

$$\mathbf{x_{opt}} = \arg\max_{\mathbf{x}} \nu(\mathbf{x}) = \arg\max_{\mathbf{x}} \frac{\mathbf{x}^T \mathbf{A}\mathbf{G}\mathbf{A}^T \mathbf{x}}{\mathbf{x}^T \mathbf{G}\mathbf{x}} \tag{8}$$

where **G** is the stationary covariance matrix of process $\mathbf{s}_t$. Based on (8), I observe that the problem is equivalent to finding the eigenvector corresponding to the maximum eigenvalue in the following generalized eigenvalue problem:

$$\mathbf{A}\mathbf{G}\mathbf{A}^T \mathbf{x} = \alpha \mathbf{G}\mathbf{x} \tag{9}$$

α can be obtained by solving the following equation:

$$\det(\mathbf{A}\mathbf{G}\mathbf{A}^T - \alpha \mathbf{G}) = 0 \tag{10}$$

2.4. Optimal sparse portfolio selection

In this section, I will define the problem of optimal portfolio selection under cardinality constraint. This corresponds to the second box in Fig. 1 and is at the very heart of the methodology. In the next section, I will review the parameter estimation techniques, represented by the first box, which will serve as input for the methods explained in this section.

In the previous section, I outlined the process of selecting a portfolio which maximizes predictability by solving a generalized eigenvalue problem. However, I did this without considering the sparseness constraint. Adding this to (8), I can formulate the constrained optimization problem as follows:

$$\mathbf{x}_{\mathbf{opt}} = \underset{\mathbf{x}\in\mathbf{R}^n, card(\mathbf{x})\le L}{\arg\max} \frac{\mathbf{x}^T\mathbf{A}\mathbf{G}\mathbf{A}^T\mathbf{x}}{\mathbf{x}^T\mathbf{G}\mathbf{x}} \tag{11}$$

where *card* denotes the number of non-zero components, and L is a given positive integer $1 \le L \le n$.

The cardinality constraint poses a serious computational challenge, as the number of subspaces in which optimality must be checked grows exponentially. In fact, Natarjan shows that this problem is equivalent to the subset selection problem, which is proven to be NP-hard [67]. However, as a benchmark metric, I can compute the theoretically optimal solution which, depending on the level of sparsity and the total number of assets, could be computationally feasible [1]. I also describe three polynomial time, heuristic algorithms for an approximate solution of this problem.

2.4.1. Exhaustive search method

The brute force approach of constructing all $\left(\frac{n!}{L!(n-L)!}\right)$ L-dimensional submatrices of $\mathbf{G}$ and $\mathbf{A}\mathbf{G}\mathbf{A}^T$ and then solving all the corresponding eigenvalue problems to find the theoretical optimum is, in general, computationally infeasible. However, for relatively small values of n and L, or as an off-line computed benchmark, this method can provide a very useful basis of comparison. Indeed, for the practical applications considered by d'Aspremont [26] (selecting sparse portfolios of n=8 U.S. swap rates and n=14 FX rates), this method is fully applicable and could be used to examine the level of sub-optimality of other proposed methods. Numerical results have shown that this method can only be applied with reasonable running time up to n=22 (see Fig. 12) which is far smaller than the portfolio size in most practical applications.

2.4.2. Greedy method

A reasonably fast heuristic algorithm is the so-called greedy method, first presented by d'Aspremont [26]. Let I_k be the set of indices belonging to the k non-zero components of $\mathbf{x}$. One can then develop the following recursion for constructing I_k with respect to k.

When $k=1$, I set $i_1 = \arg\max_{j\in[1,n]} \frac{\left(\mathbf{AGA}^T\right)_{jj}}{\mathbf{G}_{jj}}$. Suppose now that I have a reasonable approximate solution with support set I_k given by $\left(\mathbf{x}\right)_k = \arg\max_{\left\{\mathbf{x}\in R^n : x_{I_k^C}=0\right\}} \frac{\mathbf{x}^T\mathbf{AGA}^T\mathbf{x}}{\mathbf{x}^T\mathbf{Gx}}$, where I_k^C is the complement of the set I_k. This can be solved as a generalized eigenvalue problem of size *k*. I seek to add one variable with index i_{k+1} to the set I_k to produce the largest increase in predictability by scanning each of the remaining indices in I_k^C. The index i_{k+1} is then given by

$$i_{k+1} = \arg\max_{i\in I_k^C} \max_{\left\{\mathbf{x}\in R^n : x_{J_i}=0\right\}} \frac{\mathbf{x}^T\mathbf{AGA}^T\mathbf{x}}{\mathbf{x}^T\mathbf{Gx}}, \quad \text{where } J_i = I_k^C \setminus \left\{i\right\} \tag{12}$$

which amounts to solving *(n − k)* generalized eigenvalue problems of size *k + 1*. I then define $I_{k+1} = I_k \bigcup \left\{i_{k+1}\right\}$, and repeat the procedure until *k = n*. Naturally, the optimal solutions of the problem might not have increasing support sets $I_k \subset I_{k+1}$, hence the solutions generated by this recursive algorithm are potentially far from optimal. However, the cost of this method is relatively low: with each iteration costing $O\left(k^2\left(n-k\right)\right)$, the complexity of computing solutions for all target cardinalities *k* is $O\left(n^4\right)$. This recursive procedure can also be repeated forward and backward to improve the quality of the solution.

2.4.3. Truncation method

As opposed to the previous methods, a simple and very fast heuristic which I have developed is the following. First, compute $\mathbf{x}_{\mathbf{opt}}$, the unconstrained, *n*-dimensional solution of the optimization problem in (8) by solving the generalized eigenvalue problem in (9). Next, consider the *L* largest values of $\mathbf{x}_{\mathbf{opt}}$ and construct *LxL* dimensional submatrices $\mathbf{G}'$ and $(\mathbf{AGA}^T)'$ corresponding to these dimensions. Solving the generalized eigenvalue problem in this reduced space and padding the resulting $\mathbf{x}'_{\mathbf{opt}}$ with 0's will yield a feasible solution $\mathbf{x}_{\mathbf{opt}}^{trunc}$ to the original constrained optimization problem. The big advantage of this method is that with just two maximum eigenvector computations, I can determine an estimate for the optimal solution. The intuition behind this heuristic is that the most significant dimensions in the solution of the unconstrained optimization problem could provide, in most cases, a reasonable guess for the dimensions of the constrained problem. This is clearly not the case in general, but nonetheless, the truncation method has proven to be a very quick and useful benchmark for evaluating other methods.

2.4.4. Simulated annealing with random projections for constraint satisfaction over a discretized grid

In this section, I combine traditional simulated annealing methods ([48],[78]) with the sparseness constraint by applying random projection on each iteration. As a result, I am able to find a flexible and fast heuristic algorithm which implicitly satisfies the cardinality constraint. The method also has the advantage of guaranteeing to outperform the result of any other known heuristic by adding the other solution as a starting point and building in a "memory feature". Furthermore, the method can be stopped at any point and will be able to give the best solution found in the limited time window.

Another approach to solving (9) is to restrict the portfolio vector **x** to contain only integer values. Indeed, when it comes to trading methods utilizing the optimal portfolio vector, only an integer number of assets can be purchased in most markets, so this restriction more closely resembles the truth. If the per unit price of the asset is relatively large, this can give rise to a material difference to considering all real-valued portfolio vectors. As such, the equation in (11) becomes a combinatorial optimization problem over an n-dimensional discrete grid where the subspace of allowable solutions is limited to dimensionality *L*. The main difficulty in combinatorial optimization problems is that simpler algorithms such as greedy methods and local search methods tend to become trapped in local minima. A large number and variety of stochastic optimization methods have been developed to address this problem, one of which is simulated annealing as proposed by Kirkpatrick et al. [48]

At each step of the algorithm, I consider a neighboring state **w'** of the current state $\mathbf{w}_n$ and decide between moving the system to state to **w'**, or staying in $\mathbf{w}_n$. Metropolis et al [64] suggested the use of the Boltzmann-Gibbs distribution as the probability function to make this decision as follows:

$$P\left(\mathbf{w}_{n+1}=\mathbf{w}'\right)=\begin{cases}1 & \text{if} \quad \mathrm{E}(\mathbf{w}_n)>E(\mathbf{w}') \\ e^{-\frac{E(\mathbf{w}')-E(\mathbf{w}_n)}{T}} & \text{if } \mathrm{E}(\mathbf{w}_n)\le E(\mathbf{w}')\end{cases} \tag{13}$$

where *E(.)* is the function to be minimized and *T* is the temperature of the system. If *T* is fixed and the *neighbor function* which generates the random neighboring state at each step is *non-periodic* and *ergodic* (any state is reachable from any other state by some sequence of moves), then this system corresponds to a regular time-homogenous finite Markov chain. For the Metropolis transition probability function, the stationary probability distribution of finding the system at a given state **w** is given by the Boltzmann distribution:

$$\pi_{\mathbf{w}} = e^{\frac{-E(\mathbf{w})}{TK_B}} \tag{14}$$

where K_B is the Boltzmann constant, equal to $1.38 \cdot 10^{-23}$ J/K. The stationary distribution given by (14) indicates that the maximum of function $E(\mathbf{w})$ will be reached with maximum probability [78]. Furthermore, when the temperature parameter T is decreased during the procedure (also known as *cooling*), convergence in distribution to the uniform measure over globally optimal solutions has been proven by Geman and Geman [37], provided that the cooling schedule is slow enough to be at least inversely proportional to the logarithm of time. In practice, the time performance of inverse logarithmic cooling is often untenable, so faster heuristic algorithms have been developed as a result, such as exponential, geometric and adaptive cooling and applied successfully to a wide range of combinatorial optimization problems ([24],[43],[45],[46]).

In this application, based on the formulation in (8), the energy function to be minimized is defined as

$$E(\mathbf{w}) = -\frac{\mathbf{w}^T \mathbf{A}\mathbf{G}\mathbf{A}^T \mathbf{w}}{\mathbf{w}^T \mathbf{G}\mathbf{w}} \tag{15}$$

The high-level pseudocode of the general simulated annealing algorithm used is as follows:

```
s ← Greedy_sol; e ← E(s)                          // Start from Greedy
sbest ← s; ebest ← e                              // Initial "best" solution
k ← 0                                             // Energy evaluation count
reject ← 0                                        // consecutive rejections
while ~stop(reject,k,e,ebest)                     // While stop condition is
                                                  // not met
  snew ← neighbor(s)                              // Pick some neighbour
  enew ← E(snew)                                  // Compute its energy.
  if P(e, enew, temp(k/kmax)) > random() then     // Should I move to it?
    s ← snew; e ← enew;                           // Yes, change state.
    reject ← 0                                    // Reset reject counter
  else reject ← reject + 1                        // Increment reject
counter
  if enew > ebest then                            // Is this a new best?
    sbest ← snew; ebest ← enew                    // Save to 'best found'.
  k ← k + 1                                       // One more evaluation
done
return sbest                                      // Return the best found.
```

Algorithm 1. High-level pseudocode for the simulated annealing methodology used.

A very important feature of the simulated annealing method is that the **cardinality constraint can be easily built into the search**, by mapping the unconstrained **w** vector into a randomly selected L dimensional subspace on each step. The idea is that on each step, I randomly

select k of the n indices $(i_1,...,i_k); i_j \in \{1,...,n\}$ and I set the corresponding elements of **w** to be 0: $\mathbf{w}_{i_j} := 0, j = 1,...,k$.

As such, for implementing the *neighbor function*, I need a two-step approach to first randomly decide the up to L new non-zero values, and then to randomly decide the up to L new dimensions, given the starting values and dimensions.

For the selection of non-zero values, two different approaches would be to either generate them completely anew, without regard to the starting point, or to perturb the starting non-zero values by uniformly generated random values between 0 and a constant k_1. A similar choice can be made about the non-zero dimensions as well; they can either be generated completely randomly, independent of the starting point, or I can select $k_2 \le L$ random new dimensions to change within the dimensions of my starting point. Having implemented various combinations of these choices and comparing the speed of convergence and quality of solution produced on generated data, the following implementation proved to be the best from the point of view of performance (see Fig. 11)

```
function neighbor (s0, T)                          // s0 is starting point,
for i←1 upto L                                     // Generate L random
        deltas[i]=rand(0,max(1,floor(100*T))       // perturbations to values

P ← random permutation of nonzero indices in s0
Q ← random permutation of zero indices in s0
n ← rand(0,floor(L*T, (n-L)*T))                    // # dimensions to change
for i←1 upto n
   swap P[i] and Q[i]
snew=vector(P,s0[P]+deltas)                        // insert perturbed values
                                                   // into perturbed P dims
return snew
```

Algorithm 2. Pseudocode for neighbor selection in simulated annealing

The intuition behind this algorithm is that at higher temperatures more dimensions are allowed to change and by larger amounts, but as the system is cooled, only smaller changes to s0 are allowed.

2.5. Estimation of VAR (1) model parameters

As explained in the preceding sections, in the knowledge of the parameters **G** and **A**, I can apply various heuristics to approximate the L-dimensional optimal sparse mean-reverting portfolio. However, these matrices must be estimated from historical observations of the random process $\mathbf{s}_t$. This problem corresponds to the first box in Fig. 1.

While there is a vast amount of literature on the topic of parameter estimation of VAR(1) processes, recent research has focused on sparse and regularized covariance estimation ([17], [25], [75]). However, my novel approach will be to find an unconstrained estimate for **G** which best describes the observed historical data and to deal with dimensionality reduction with the more sophisticated apparatus outlined above. Another important objective that I pose for the parameter fitting is to provide a measure of model fit of the real time series to VAR (1) model, which I can use in the portfolio selection and trading parts of my overall algorithm.

2.5.1. Estimation of matrix A

Recall from my earlier discussion that I assume $\mathbf{s}_t$ follows a stationary, first order autoregressive process as in equation (5). I first observe that if the number of assets n is great than or equal to the length of the observed time series, then A can be estimated by simply solving the linear system of equations:

$$\hat{\mathbf{A}}\mathbf{s}_{t-1} = \mathbf{s}_t. \tag{16}$$

This gives a perfect VAR(1) fit for the time series in cases where there is a large portfolio of potential assets (e.g. considering all 500 stocks which make up the S&P 500 index), from which a sparse mean-reverting subportfolio is to be chosen.

In most practical applications, however, the length of the available historical time series is greater than the number of assets considered, so **A** is estimated using, for example, least squares estimation techniques, as in

$$\hat{\mathbf{A}} : \min_{\mathbf{A}} \sum_{t=2}^{T} \left\| \mathbf{s}_t - \mathbf{A}\mathbf{s}_{t-1} \right\|^2 \tag{17}$$

where $\left\| \cdot \right\|^2$ denotes the Euclidian norm.

Equating to zero the partial derivatives of the above expression with respect to each element of the matrix **A**, I obtain the following system of equations:

$$\sum_{k=1}^{n} \hat{\mathbf{A}}_{i,k} \sum_{t=2}^{T} s_{t-1,k} s_{t-1,j} = \sum_{t=2}^{T} s_{t,i} s_{t-1,j} \; \forall i, j = 1, \ldots, n. \tag{18}$$

Solving for $\hat{\mathbf{A}}$ and switching back to vector notation for **s**, I obtain

$$\hat{\mathbf{A}} = \sum_{t=2}^{T} (\mathbf{s}_{t-1}^T \mathbf{s}_{t-1})^+ (\mathbf{s}_{t-1}^T \mathbf{s}_t), \tag{19}$$

where $(\mathbf{s}_{t-1}^T \mathbf{s}_{t-1})^+$ denotes the Moore-Penrose pseudoinverse of matrix $(\mathbf{s}_{t-1}^T \mathbf{s}_{t-1})$. Note that the Moore-Penrose pseudoinverse is preferred to regular matrix inversion, in order to avoid problems which may arise due to the potential singularity of $\mathbf{s}_{t-1}^T \mathbf{s}_{t-1}$.

2.5.2. Estimation of the covariance matrix of W

Assuming that the noise terms in equation (1) are i.i.d. with $\mathbf{W}_t \sim N(0,\sigma I)$ for some $\sigma > 0$, I obtain the following estimate for σ using $\hat{\mathbf{A}}$ from (19):

$$\hat{\sigma} = \sqrt{\frac{1}{n(T-1)}\sum_{t=2}^{T}\left\|\mathbf{s}_t - \hat{\mathbf{A}}\mathbf{s}_{t-1}\right\|^2}. \tag{20}$$

In the case that the terms of $\mathbf{W}_t$ are correlated, I can estimate the covariance matrix **K** of the noise as follows:

$$\hat{\mathbf{K}} = \frac{1}{(T-1)}\sum_{t=2}^{T}(\mathbf{s}_t - \hat{\mathbf{A}}\mathbf{s}_{t-1})^T(\mathbf{s}_t - \hat{\mathbf{A}}\mathbf{s}_{t-1}). \tag{21}$$

This noise covariance estimate will be used below in the estimation of the covariance matrix.

2.5.3. Estimation of covariance matrix G

There are two independent approaches to estimating the covariance of a VAR(1) process based on a sample time series. On the one hand, the sample covariance and various maximum likelihood-based regularizations thereof can provide a simple estimate and have been studied extensively for the more general case of multivariate Gaussian distributions ([17],[25],[27],[75]). In this treatment, I take the approach of using the sample covariance matrix directly without any regularization or finding structure via maximum likelihood, as sparsifying and structure finding will be left for the more sophisticated apparatus of the sparse portfolio selection, explained in Section 2. As such, I will define $\hat{\mathbf{G}}_1$ as the sample covariance defined as

$$\hat{\mathbf{G}}_1 := \frac{1}{T-1}\sum_{t=1}^{T}\left(\mathbf{s}_t - \bar{\mathbf{s}}\right)^T\left(\mathbf{s}_t - \bar{\mathbf{s}}\right), \tag{22}$$

where $\bar{\mathbf{s}}$ is the sample mean vector of the assets defined as

$$\bar{\mathbf{s}} := \frac{1}{T}\sum_{t=1}^{T}\mathbf{s}_t. \tag{23}$$

On the other hand, starting from the description of the VAR(1) process in (5) and assuming the more general case that the terms of $\mathbf{W}_t$ are correlated with covariance matrix **K,** I must have

$$\mathbf{G}_t = \mathbf{A}\mathbf{G}_{t-1}\mathbf{A}^T + \mathbf{K}, \tag{24}$$

which implies that in the stationary case

$$\mathbf{G} = \mathbf{A}\mathbf{G}\mathbf{A}^T + \mathbf{K}. \tag{25}$$

Having estimated **A** and **K,** as in the previous sections, this is a Lyapunov equation with unknown **G** and can be solved analytically to obtain an independent covariance estimate $\hat{\mathbf{G}}_2$. One potential issue with this approach is that $\hat{\mathbf{G}}_2$ is not necessarily positive definite and, as such, it may not be a permissible covariance estimate. This situation arises due to the fact that matrix **A** has been estimated using least squares best fit, without restricting it to be positive-definite and all of its eigenvalues to have modulus smaller than 1. Therefore, the stability theorem guaranteeing the positive definiteness of **G** cannot be used. In order to overcome this issue, in case the solution of the Lyapunov equation is non-positive-definite, the following iterative numerical method can be used to obtain a permissible covariance estimate $\hat{\mathbf{G}}_2$:

$$\mathbf{G}(k+1) = \mathbf{G}(k) - \delta(\mathbf{G}(k) - \mathbf{A}\mathbf{G}(k)\mathbf{A}^T - \mathbf{K}), \tag{26}$$

where δ is a constant between 0 and 1 and $\mathbf{G}(k)$ is the covariance matrix estimate on iteration *k.* Provided that the starting point for the numerical method, **G**(0), is positive definite (eg. the sample covariance matrix) and since the estimate of **K** is positive definite, by construction, this iterative method will produce an estimate which will be positive definite. It can also be seen that with appropriate choice of δ and stopping condition, this numerical estimate will converge to the solution of the Lyapunov equation in (25), in case that is positive definite.

In latter sections some numerical results are presented which show that for generated VAR(1) data, these two covariance estimates are equivalent, provided that appropriately sized sample data is available for the given level of noise. However, for historical financial data, the two estimates can vary significantly. A large difference between the two estimates indicates a poor fit of the data to the VAR(1) model, hence I can define the following measure of model fit:

$$\beta := \left\| \hat{\mathbf{G}}_1 - \hat{\mathbf{G}}_2 \right\|, \tag{27}$$

where $\|\mathbf{M}\|$ denotes the largest singular value of matrix **M.**

2.6. Trading as a problem of decision theory

Having identified the portfolio with maximal mean reversion that satisfies the cardinality constraint, the task now is to develop a profitable convergence trading strategy. The immediate decision that we face is whether the current value of the portfolio is below the mean and is therefore likely to rise so buying is advisable, above the mean and therefore likely to fall so selling is advisable, or close to the mean in an already stationary state, in which case no action will likely result in a profit. In order to formulate this as a problem of decision theory, I first need to estimate the mean value of the portfolio. In our overall framework, depicted in Fig. 1, this corresponds to the third box entitled “Portfolio parameter identification” and is a critically important step on the way to developing a profitable trading strategy. Note that once the mean is identified, we can move on to

the fourth box in Fig. 1 and identify a *trading strategy* based on buying sparse mean reverting portfolios below the estimated mean and selling above the estimated mean. A simple, binary model of trading in which we restrict ourselves to holding only 1 portfolio at a time can be perceived as a walk in a binary state-space, depicted in Fig. 2.

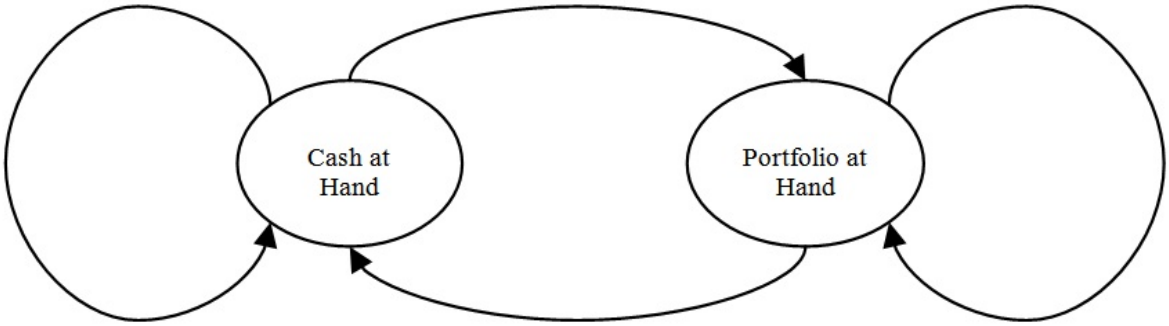

Fig. 2: Trading as a walk in a binary state-space

In practice, there are much more complex models of trading, but it proves to be very interesting to study this approach and the results allow some more general conclusions to be drawn. I present three different methods of identifying μ from the observations of the optimal portfolio's time series, $\{\mathbf{p}_t, t = 1,...,m\}$.

2.6.1. Sample Mean estimator

The simplest estimate of the mean of the process based on the time series observation of the historical values of the portfolio is the one given by the sample mean, defined as follows:

$$\hat{\mu}_1 := \frac{1}{m}\sum_{t=1}^{m}\mathbf{p}_t \tag{28}$$

This estimate is akin to a measure used by technical traders and gives a poor estimate of the mean-reverting process, in case it is in its transient state, trending towards the mean. However, it has served as a very useful benchmark and performs surprisingly well in terms of trading results, as we will see later.

2.6.2. Mean Estimation via least squares linear regression

Following the treatment of Smith [83], I can perform a linear regression on pairwise consecutive samples of **p** as follows:

$$\mathbf{p}_{t+1} = a\mathbf{p}_t + b + \varepsilon \tag{29}$$

for $t = 1,...,m-1$.

Then, an estimate of μ can be obtained from the coefficients of the regression as follows:

$$\hat{\mu}_2 := \frac{b}{1-a} \tag{30}$$

Note that there is an instability in this estimate for a=1 in which case $\hat{\lambda} = -\ln a = 0$ and hence the process is deemed not to be mean reverting based on the observed sample.

2.6.3. Mean Estimation via pattern matching

In the description of this novel mean estimation technique, I start from the definition of the discrete Orstein-Uhlenbeck process in equation (1) and consider its continuous solution given in equation (2). Disregarding the noise to consider only the expected value of the process, I can rewrite equation (3) as follows:

$$\mu(t) = \mu + \left(\mu(0) - \mu\right)e^{-\lambda t} \tag{31}$$

where $\mu(t) = E\left[\, p(t)\right]$ is a continuous process of the expected value of the portfolio at time step t with $\mu(0) = p(0)$. Intuitively, this describes the value of the portfolio, without noise, in the knowledge of the long term mean and the initial portfolio value.

In Fig. 3, I show some typical tendencies of $\mu(t)$ for various relative values of $\mu(0)$ and μ. The idea behind pattern matching is to observe historical time series values of $\mathbf{p}_t = \left(p(t-1), p(t-2),..., p(0)\right)$ and use maximum likelihood estimation techniques to determine which of the patterns $\mu(t)$ matches best.

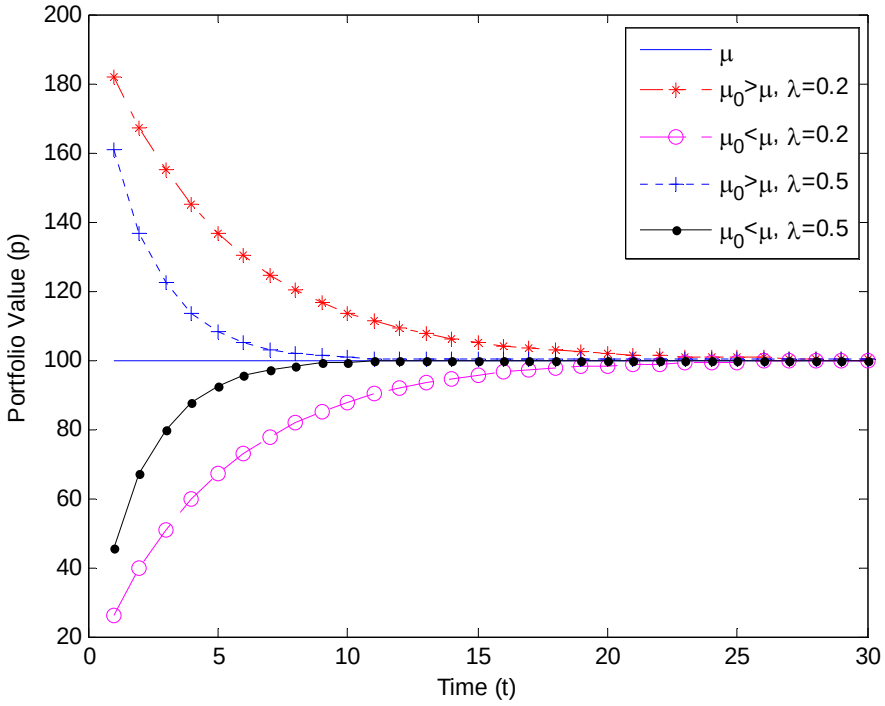

Fig. 3: Some sample patterns of $\mu(t)$

I first observe that $\mathbf{p}_t = \left(p(t-1), p(t-2), ..., p(0)\right)$ is subject to a multivariate Gaussian distribution, the density function of which is given by

$$D(\mathbf{p}_t) = \frac{1}{\sqrt{(2\pi)^t \det(\mathbf{U})}} e^{-\frac{1}{2}(\mathbf{p}_t - \boldsymbol{\mu}_t)^T \mathbf{U}^{-1}(\mathbf{p}_t - \boldsymbol{\mu}_t)} \tag{32}$$

where $\mathbf{U}_{ij} := E\left(p(t-i)\,p(t-j)\right) = \frac{\sigma^2}{2\lambda} e^{-\lambda(i+j)}\left(e^{2\lambda i} - 1\right)$ is the time-correlation matrix of $\mathbf{p}_t$ and $\boldsymbol{\mu}_t = \left(\mu(t-1), ..., \mu(0)\right)$. The idea is now to consider class Φ of all t-length vectors, $\boldsymbol{\mu}_t$ which consist of sequences satisfying (31). The maximum likelihood "pattern matching" estimate can now be formulated as follows:

$$\hat{\boldsymbol{\mu}}_{\mathbf{t}} = \arg\max_{\boldsymbol{\mu}_t \in \Phi} \frac{1}{\sqrt{(2\pi)^t \det(\mathbf{U})}} e^{-\frac{1}{2}(\mathbf{p}_t - \boldsymbol{\mu}_t)^T \mathbf{U}^{-1}(\mathbf{p}_t - \boldsymbol{\mu}_t)}. \tag{33}$$

This is equivalent to

$$\hat{\boldsymbol{\mu}}_{\mathbf{t}} = \arg\min_{\boldsymbol{\mu}_t \in \Phi} \left\{ \boldsymbol{\mu}_t^T \mathbf{U}^{-1} \boldsymbol{\mu}_t - 2\boldsymbol{\mu}_t^T \mathbf{U}^{-1} \mathbf{p}_t \right\}. \tag{34}$$

Using the definitions of $\mathbf{U}_{ij}$ and $\boldsymbol{\mu}_t$ to expand (34) then taking the derivative of this quadratic expression with respect to μ, equating to zero and solving, I obtain the following closed-form estimate for the long term mean:

$$\hat{\mu}_3 := \frac{\sum_{i=1}^{t}\sum_{j=1}^{t}\left(\mathbf{U}^{-1}\right)_{i,j}\left[\mu(0)\left(2e^{-\lambda(i+j)} - e^{-\lambda i} - e^{-\lambda j}\right) - 2\mathbf{p}_j\left(e^{-\lambda i} - 1\right)\right]}{\sum_{i=1}^{t}\sum_{j=1}^{t} 2\left(\mathbf{U}^{-1}\right)_{i,j}\left(e^{-\lambda(i+j)} - e^{-\lambda i} - e^{-\lambda j} + 1\right)}. \tag{35}$$

Having observed the historical portfolio values in $\mathbf{p}_t$ and $\mu(0) = \mathbf{p}_0$, I can substitute $\sigma = \mathbf{x}^T\hat{\mathbf{K}}\mathbf{x}$ and λ computed via linear regression [83] into this equation to obtain an estimate for the long term mean.

2.6.4. A simple *convergence trading* strategy

The main task after identifying the mean reverting portfolio and obtaining an estimate for its long-term mean μ, is to verify whether $\mu(t) < \mu$ or $\mu(t) \geq \mu$ based on observing the samples $\{p(t) = \mathbf{x}^T\mathbf{s}(t), t = 1,...,T\}$. This verification can be perceived as a decision theoretic problem, since direct observations of $\mu(t)$ are not available.

If process $p(t)$ is in stationary state then the samples $\{p(t), t = 1,...,T\}$ are generated by a Gaussian distribution $N\left(\mu, \sqrt{\frac{\sigma^2}{2\lambda}}\right)$. As a result, for a given rate of acceptable error ε, we can select an α for which

$$\int_{\mu-\alpha}^{\mu+\alpha} \frac{1}{\sqrt{2\pi\sigma^2/2\lambda}} e^{-\frac{(u-\mu)^2}{\sigma^2/\lambda}} du = 1 - \varepsilon \tag{36}$$

.

As such, having observed the sample $p(t) \in [\mu - \alpha, \mu + \alpha]$, it can be said that we accept the stationary hypothesis which holds with probability $1-\varepsilon$. Thus the trading strategy can then be summarized as follows:

- If the observed sample $p(t) < \mu - \alpha$ then we accept the hypothesis that $\mu(t) < \mu$ The error probability of this hypothesis is $\int_{-\infty}^{\mu-\alpha} \frac{1}{\sqrt{2\pi\sigma^2/2\lambda}} e^{-\frac{(u-\mu)^2}{\sigma^2/\lambda}} du = \varepsilon/2$. In this case, we buy the portfolio in case we have cash at hand and we hold the portfolio if we already have one.
- If the observed sample $p(t) > \mu + \alpha$ then we accept the hypothesis that $\mu(t) > \mu$ The error probability of this hypothesis is $\int_{\mu+\alpha}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2/2\lambda}} e^{-\frac{(u-\mu)^2}{\sigma^2/\lambda}} du = \varepsilon/2$. In this case, if we have a portfolio, we sell it, otherwise we perform no action.

- If the observed sample $p(t) \in [\mu - \alpha, \mu + \alpha]$ then we accept the hypothesis that $\mu(t) = \mu$ The error probability of this hypothesis is $1 - \int_{\mu-\alpha}^{\mu+\alpha} \frac{1}{\sqrt{2\pi\sigma^2/2\lambda}} e^{-\frac{(u-\mu)^2}{\sigma^2/\lambda}} du = \varepsilon$. Then sell if a portfolio is held, or perform no action if only cash is held.

As such, I can now extend Fig. 2 to present a complete flowchart for the proposed simple trading strategy in Fig. 4.

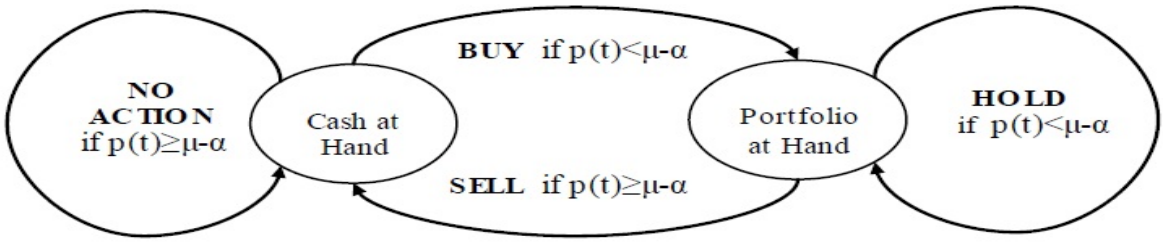

Fig. 4: Flowchart for simple convergence trading of mean reverting portfolios

2.6.5. Some financial and practical considerations of trading

Having worked out the mathematical and algorithmic foundations for mean reverting trading, in this section some of the financial considerations are examined which arise during trading. One important question is whether to spend all the available cash at the time we identify a mean-reverting portfolio which is below its long-term mean level or to use some *cash management* strategy to only spend part of the current holdings. Indeed, we have two parameters, β and λ which we can use to help establish the level of confidence in the profitability of the portfolio. For simplicity, in the presented results I have used the approach of spending all of the available cash each time I identify an appropriate portfolio.

Furthermore, the simple convergence trading strategy I described in the previous section allows holding only one portfolio at a time. I could enhance this to have the ability to hold a number of portfolios at once, continually estimating the remaining profitability of each and comparing this to the best available portfolio in the market at each time step.

Finally, in the numerical results, I have assumed that we have the ability to buy and sell assets without any transaction costs and we also have the ability to short sell. In order to make the results more realistic, I could introduce a bid-ask spread or a more sophisticated order book model to estimate the true profitability of these methods in the presence of market friction.

In terms of the applicability of the methodology in real life, two different approaches may be taken. Firstly, by applying methods which can be implemented in real-time with sub-second

response time, the convergence trading methodology can be implemented in a high frequency trading setting. With the arrival of each new tick in the market, the VAR(1) model parameters can be re-estimated and the optimal portfolio selection algorithm can be run. Depending on the positioning of the price of the portfolio with respect to its long term mean, the portfolio may be purchased. In case we already hold a portfolio, its long term mean should be recalculated and the appropriate action should be taken based on this. Given the necessary computation times, this can only be realistically done in real-time with the truncation or greedy methods. Alternatively, in a low frequency trading environment, where new information from the market is only observed on a daily or hourly basis, more complex methods such as simulated annealing, has enough time to run and suggest the optimal portfolio to be purchased. As shown, this method consistently outperforms the simpler portfolio selection methods and therefore should result in portfolios with higher mean reversion.

2.7. Performance analysis of sparse portfolio optimization

In this section, I will review some results of the numerical tests which were produced for validating the methods outlined earlier. I first tested the model parameter estimation methods on generated data to show their viability and observe their limitations. I then compared the effectiveness of the greedy search to the theoretically optimal exhaustive method on generated time series.

2.7.1. Performance of VAR(1) model parameter estimation

In order to compare the portfolio selection methods outlined in section 2, I generated synthetic VAR(1) data as follows. I first generated an $n \times n$ random matrix $\mathbf{A}$ ensuring that all its eigenvalues were smaller than 1. I then generated random i.i.d. noise $\mathbf{W}_t \sim N(0, \sigma I)$ for an arbitrary selection of σ. Finally, I used equation (5) to generate the random sequence $\mathbf{s}_t$ ensuring that all its values were positive Then, for increasing sizes of *t*, I ran a number of independent tests and looked at the average error in norm. Fig. 5 shows that the estimate gets progressively better with the size of the available time series and that it is remarkably resilient to increasing values of σ.

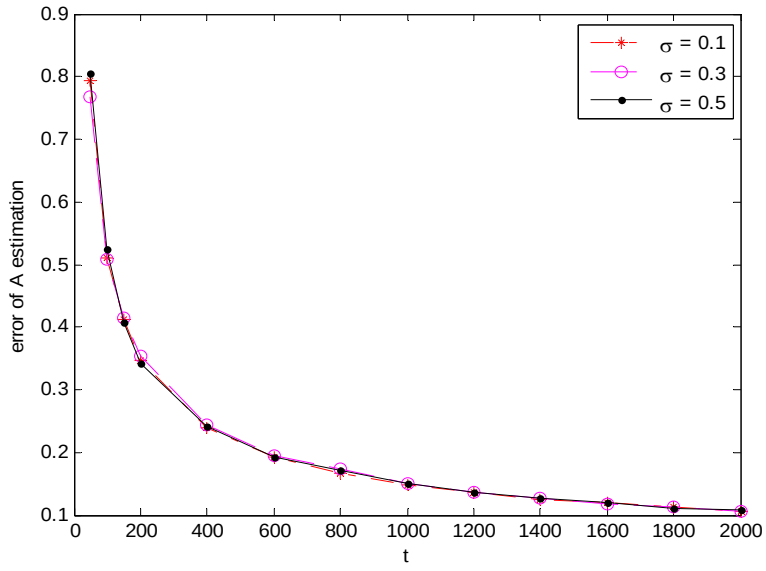

Fig. 5: $\left\|\hat{\mathbf{A}}-\mathbf{A}\right\|$ vs. t for n=8, σ=0.1, 0.3, 0.5, generating 100 independent time series for each t and plotting the average norm of error

In a similar fashion, I compared the covariance estimate $\hat{\mathbf{G}}_1$ as defined in (22) to $\hat{\mathbf{G}}_2$, the solution of the Lyapunov equation, substituting $\hat{\mathbf{A}}$ and $\hat{\sigma}$ as estimated from the data into equation (25). I then considered the proximity of these two estimates on increasing sample sizes and increasing values of σ, taking the average of a number of independent tests. Fig. 6 shows the results.

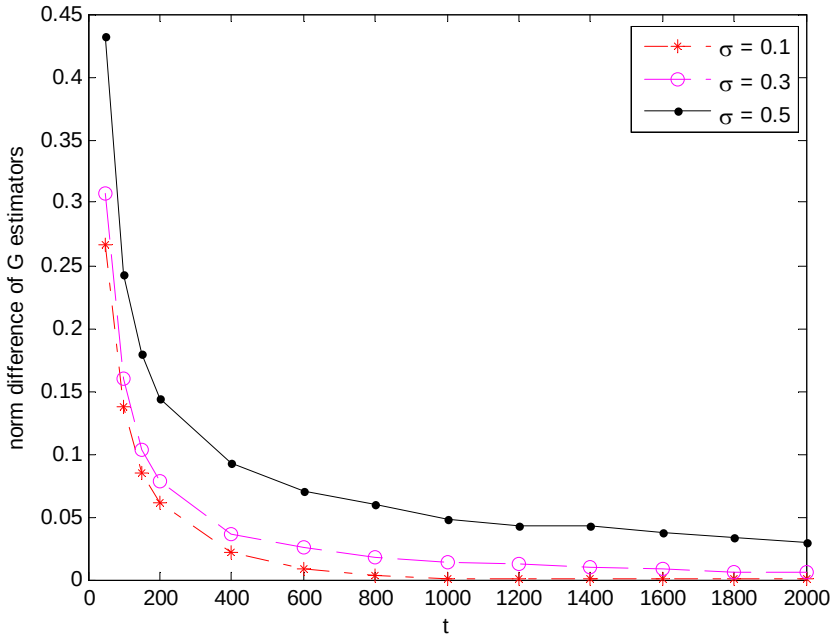

Fig. 6: $\left\|\hat{\mathbf{G}}_1-\hat{\mathbf{G}}_2\right\|$ vs. t for n=8, σ=0.1, 0.3, 0.5, generating 100 independent time series for each t and plotting the average norm of error

It can be seen that there is good agreement between the two estimates of covariance even for relatively small amounts of data (100-300 data points) which gets increasingly better with the growth of the sample size. Note also that there is significant difference between the different amounts of noise on the process, smaller noise implies a better agreement between the two covariance indicators. In order to gain a better understanding of the convergence of the two covariance estimators, we examined the relative difference $\frac{\left\|\hat{\mathbf{G}}_1 - \hat{\mathbf{G}}_2\right\|}{\left\|\hat{\mathbf{G}}_1\right\|}$ for varying levels of σ. The results, shown on Fig. 7, show good convergence of the two norms for all values of σ.

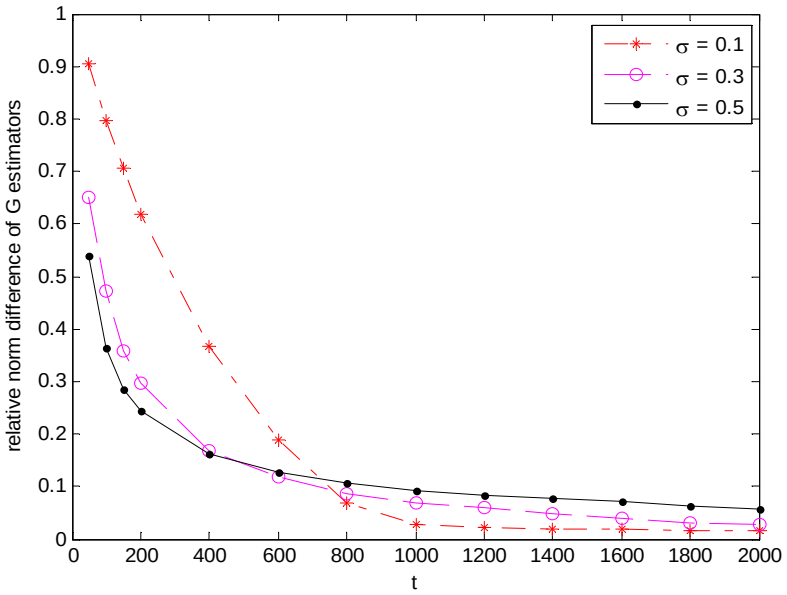

Fig. 7: $\frac{\left\|\hat{\mathbf{G}}_1 - \hat{\mathbf{G}}_2\right\|}{\left\|\hat{\mathbf{G}}_1\right\|}$ vs. t for n=8, σ=0.1, 0.3, 0.5, generating 100 independent time series for each t and plotting the relative difference of covariance estimators

In the next numerical test, I used fixed sized sliding windows of various sizes over the generated VAR(1) sequence and I compared the sample covariance in the current time window with the current value of the recursive relation in (24). Fig. 8 shows the values of the two estimates for varying levels of σ. I can see that both measures converge to the same level, but the covariance estimate obtained from (24) is more resistant to noise in the sample data. Note that all results were generated with a fixed $\delta = 0.1$, which guaranteed convergence and was used consistently throughout my numerical results. These results could be further optimized via empirical analysis of the accuracy and convergence of the numerical estimator with respect to this δ parameter which is left as an area of future research.

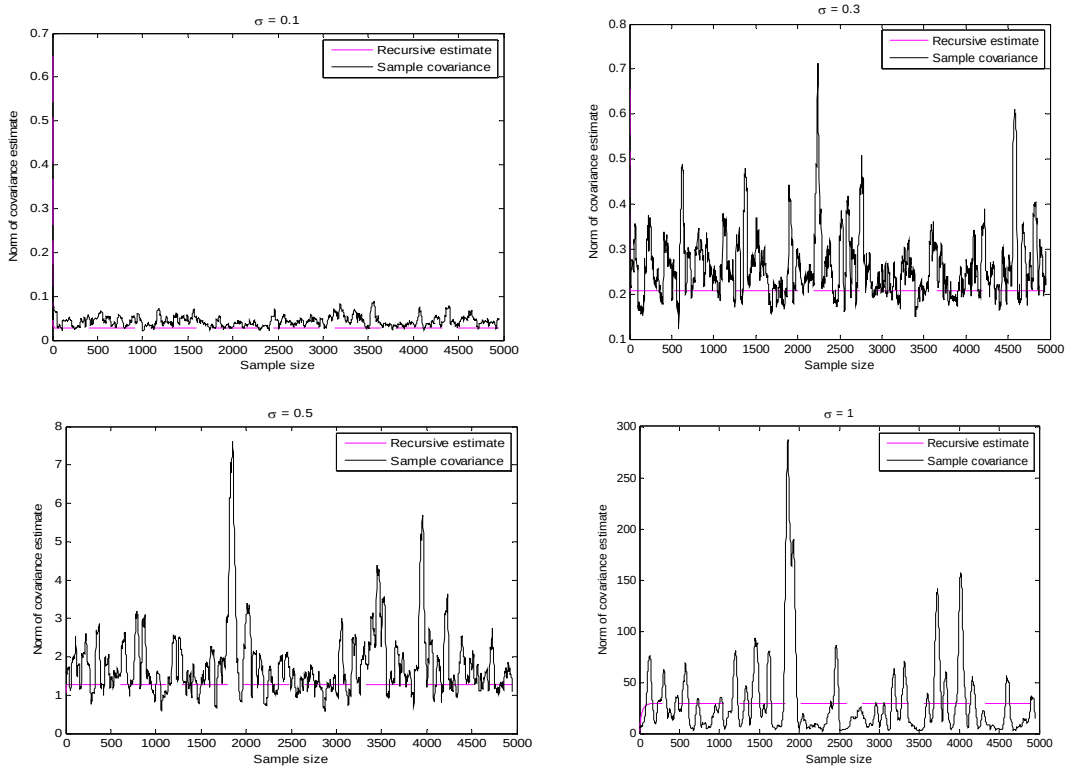

Fig. 8: Sample covariance and recursive covariance estimates over sliding windows of size 50 over 5000 samples for σ=0.1, 0.3, 0.5, 1 (note the differences in scaling of the plots)

2.7.2. Performance of Ornstein-Uhlenbeck model parameter estimation

In order to test the relative accuracy of the three different estimates of μ presented in Chapter 4, I used simulated Ornstein-Uhlenbeck processes. Note that for simulation purposes, as shown in [38] instead of equation (1), it is more appropriate to use the discrete version of the differential equation, given by

$$p(t) = e^{-\lambda \Delta t} p(t-1) + \left(1 - e^{-\lambda \Delta t}\right)\mu + \sigma \sqrt{\frac{\left(1 - e^{-2\lambda \Delta t}\right)}{2\lambda}} dW(t) \tag{37}$$

As seen in Fig. 9, for a reasonably sized λ and sufficiently large sample, the linear regression method produces a smaller Mean Squared Error (MSE) than the other two estimates.

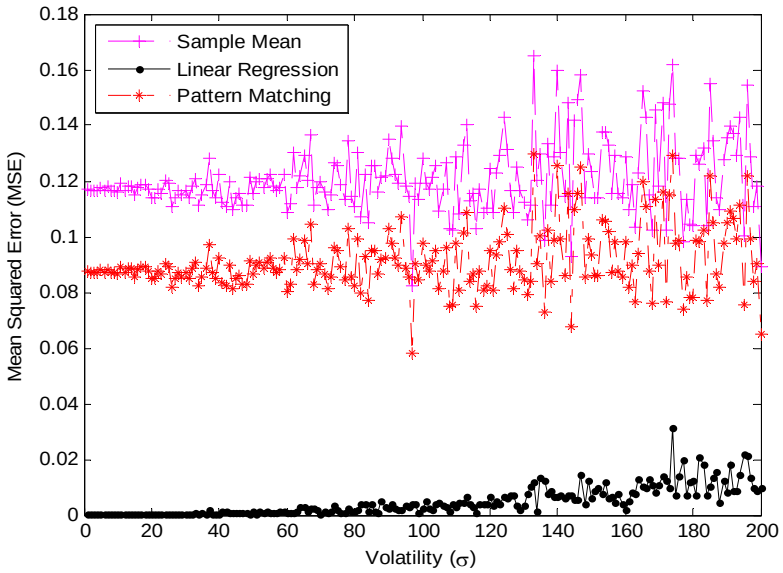

Fig. 9: MSE of each μ estimate as a function of σ. Fixing $\lambda = 1$, $\mu_0 = 0$ and sample size of 100, I generated noisy sequences as per (37) with $\mu = 0,1,2,...,50$ and $\sigma = 0.01, 0.02, ..., 2.00$.

On the other hand, Fig. 10 shows that the linear regression method is unstable for smaller sample size (in these experiments, I used 20) and for small values of λ. Note also that as I increase λ, the pattern matching estimate produces increasing MSE with the limit of producing identical MSE to the sample mean method.

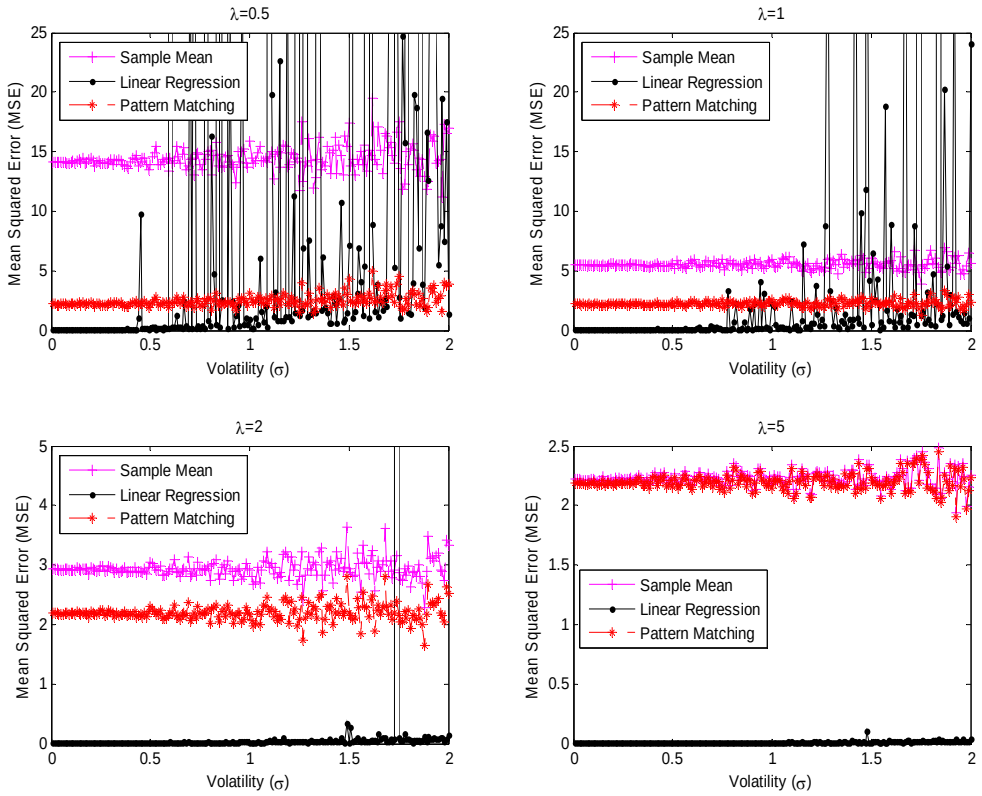

Fig. 10: MSE of each μ estimate as a function of σ. λ=0.5, 1, 2, 5, $\mu_0 = 0$ and sample size of 20.

2.7.3. Performance of portfolio selection and trading on generated data

In order to compare the portfolio selection methods outlined in section 2.4, I generated synthetic VAR(1) data as follows. I first generated an $n \times n$ random matrix **A,** ensuring that all of its eigenvalues were smaller than 1. I then generated random i.i.d. noise $\mathbf{W}_t \sim N(0, \sigma I)$ for an arbitrary selection of σ. Finally, I used equation (5) to generate the random sequence $\mathbf{s}_t$ ensuring that all of its values were positive. I then used the methods of section 3 to compute the estimates $\hat{\mathbf{A}}$, $\hat{\mathbf{K}}$ and $\hat{\mathbf{G}}$, and computed optimal sparse portfolios, maximizing the mean-reversion coefficient λ.

I will now provide some technical details for the implementation of the simulated annealing algorithm. As explained in section 2.4.4, optimality in distribution of the solution of simulated annealing can be proven, as long as the *cooling schedule* is slow enough to be at least inversely proportional to the logarithm of time [37]. However, for concrete applications this schedule is "utterly impractical […] and amounts to a random search in state space." [69] As such, I implemented the exponential cooling schedule of the form

$$T(t) = T_0 \alpha^t \tag{38}$$

In my implementation, I used T_0=0, α=0.8 and a maximum of 3,000 repeats at each temperature level, but moving to the next temperature when 100 successful moves have been made. This technique is used to ensure there are a sufficient number of trials at each temperature level, but also enables the algorithm to proceed if there is reasonable progress.

The same principles must guide my selection of *stopping conditions*. Since my specific application contains no *a priori* target or lower limit for the minimization problem, I set a stopping temperature $T_{stop}=10^{-8}$. However, if the algorithm finds no improvement over 10,000 consecutive trials, I also stop the algorithm. Another optimization heuristic I found successful for difficult surfaces is to revert to the best solution thus far after 500 unsuccessful attempts.

As shown, it is practical to select a competitive *starting point* for the algorithm because, by my construction, the final solution is guaranteed to be at least as good as the starting point. Given the linear scalability of eigenvectors, I can give any scalar multiple of the solution of the greedy algorithm as the starting point for the algorithm. Given that it only operates over discrete whole values, it has been found advisable to give a large scalar multiple of the normalized greedy solution as a starting point – for the purposes of my numerical runs, I used 100 times the normalized greedy solution.

Having run 3,000 independent simulations for selecting sparse portfolios of five assets out of a universe of ten, I found that the greedy method generates the theoretically best result produced by an exhaustive search in 70% of the cases. Of the remaining 30% where an improvement over the greedy method is possible, simulated annealing managed to find an improvement in slightly over

one-third of the cases, namely in 11% of all simulations. The impact of the improvement produced by the simulated annealing method can be significant, as illustrated in Fig. 11 by one specific generated example where simulated annealing substantially outperforms the greedy method. I also note that the truncation method performs poorly in this analysis, providing mean reversion coefficients lower than other methods in over 99% of the generated cases.

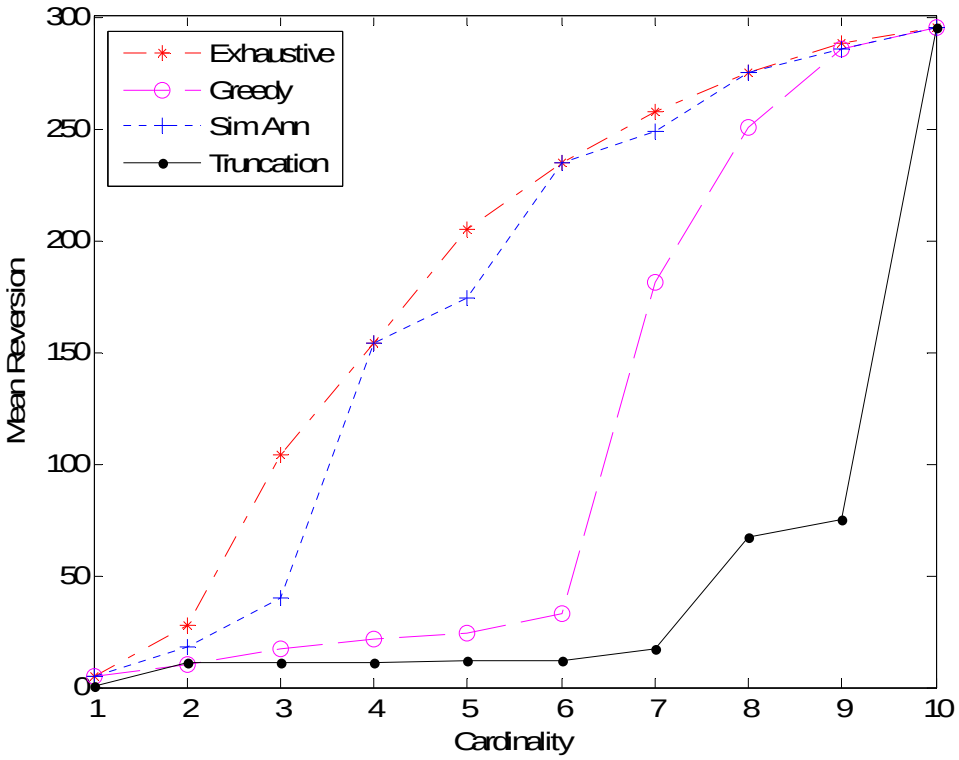

Fig. 11: Comparison of portfolio selection methods of various cardinalities on n=10-dimensional generated VAR(1) data.

In the following simulation, I increased the asset population to 20 and restricted cardinality of the optimal portfolio to ten. Intuitively, this causes the greedy method to go astray more frequently and thus would provide more room for improvement. The simulations indeed confirm this intuition, as in a simulation of 1,000 independent cases, the greedy method was only able to find the theoretical optimum in only 1% of cases, with simulated annealing outperforming the greedy method in 25% of all cases. This indicates that for larger problem sizes, simulated annealing becomes more attractive compared to other simple heuristic methods. This finding is also confirmed when analyzing the runtimes of the different algorithms. As expected, truncation and greedy methods are clearly the fastest of the four examined methods. The speed of simulated annealing depends largely on the settings of the parameters, but it is generally slower than even an exhaustive search for smaller values of n. However, over $n=21$, simulated annealing is faster than exhaustive search. At the other end of the spectrum, the greedy and truncation methods, although clearly less optimal than exhaustive and simulated annealing, are very fast and therefore can be used for real-

time algorithmic trading applications for most reasonably-sized problems. For a total asset size of 100, to compute all sparse portfolios of cardinalities one to 100 took only two seconds with the truncation method and only 31 seconds with the greedy algorithm on a Pentium 4, 3.80 GHz machine. Simulated annealing can be used for lower frequency (daily or infrequent intraday trading) for most problem sizes. The optimal exhaustive method is only practical for daily or intraday trading if the total number of assets does not exceed 25. Fig. 12 shows more details of the runtimes of the different algorithms.

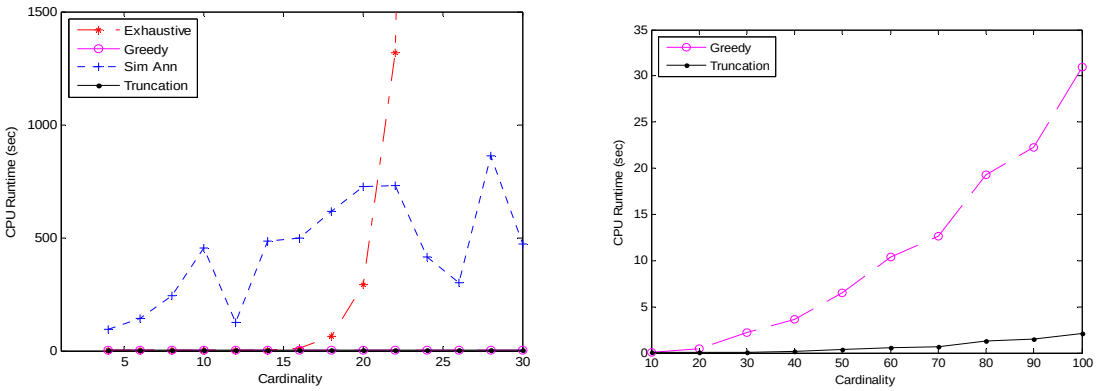

Fig. 12: CPU runtime (in seconds) versus total number of assets *n,* to compute a full set of sparse portfolios, with cardinality ranging from 1 to *n,* using the different algorithms.

In order to prove the economical viability of the methodology, I implemented the simple convergence trading strategy, as outlined in section 4.3. I generated n=10-dimensional VAR(1) sequences of length 270, of which the first 20 time steps were used to estimate parameters of the model and find the optimal portfolio of size L=5 using the different methods. The following 250 (approximate number of trading days in a year) were used to trade the portfolio, using the simple linear regression estimate of μ. Running each of the algorithms on 2,000 different time series, I found that all methods generated a profit in over 97% of the cases. The size of the profit, starting from \$100, using the risky strategy of betting all of the cash on each opportunity, increased monotonically in most simulations, reaching as high as 400% in some cases. The biggest loss across the 2,000 runs was 37% of the initial wealth, showing the robustness of the method on generated data. Fig. 13 shows a typical pattern of successful convergence trading on mean-reverting portfolios selected from generated VAR(1) data.

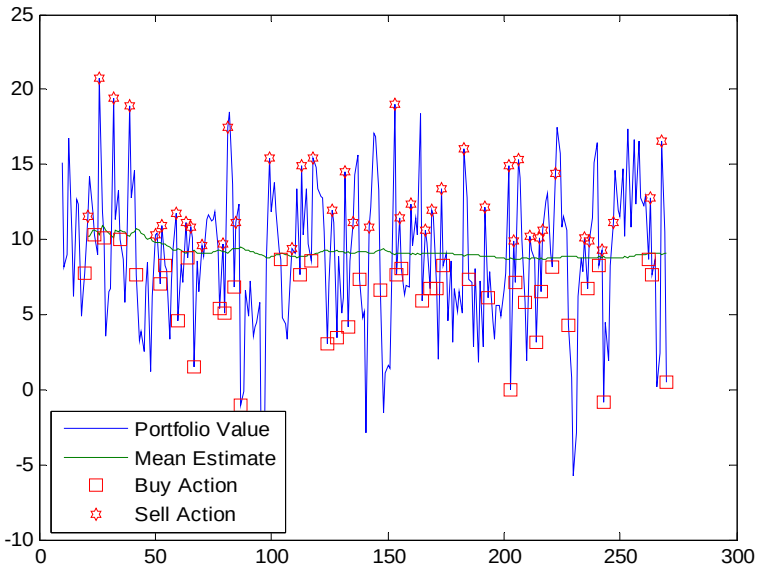

Fig. 13: Typical example of convergence trading over 250 time steps on a sparse mean-reverting portfolio. Weighted mix of L=5 assets were selected from n=10-dimensional generated VAR(1) process by simulated annealing during the first 20 time steps. A profit of $1,440 was achieved with an initial cash of $100 after 85 transactions.

I observe that the more frequent the movement around the estimated long-term mean, the more trading opportunities arise, hence the larger the profitability of the methodology. Fig. 14 shows the histogram of trading gains achieved by the simulated annealing method.

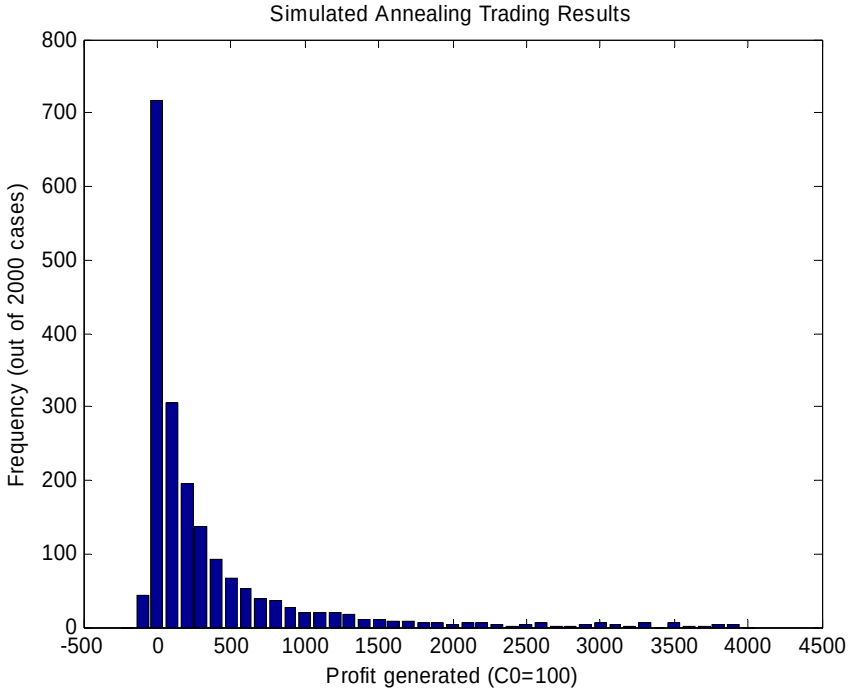

Fig. 14: Histogram of profits achieved over 2,000 different generated VAR(1) series by simulated annealing.

All four methods produced average profits in the same order of magnitude, with the distribution of trading gains very similar. This is despite the fact that the exhaustive method produced mean reversion coefficients on average 15 times those produced by the truncation method, and three times those produced by the greedy method and simulated annealing. This implies that the profits reached by this simple convergence trading strategy are not directly proportional to the lambda produced by the portfolio selection method. In order to maximize trading gains, other factors (such as the model fit, the amount of diversion from the long-term mean, etc) would need to be taken into account. This topic is the subject of further research.

2.7.4. Performance of portfolio selection and trading on historical data

In accordance with the related literature ([5],[26],[73]), I consider daily close prices of the 500 stocks which make up the S&P 500 index from July 2009 to July 2010. I use the methods of section 2.5 to estimate the model parameters on a sliding window of eight observations and select sparse, mean reverting portfolios using the algorithms of section 2.4. Using the simple convergence trading methodology in section 2.6.4 for portfolios of sparseness L=3 and 4, the methods produced annual returns in the range of 23-34% (note that the return on the SP500 index was 11.6% for this reference period). Simulated annealing performed similarly to the greedy method for most cardinalities, but produced better mean reversion coefficients and better profit return for the case L=4 (see Fig. 15).

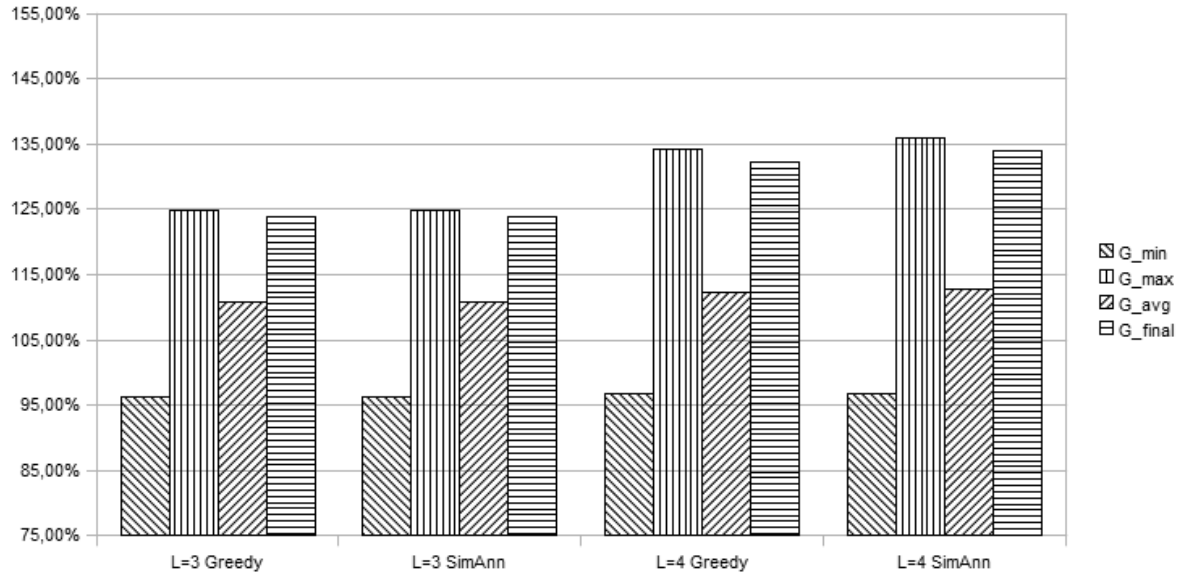

Fig. 15: Comparison of minimum, maximum, average and final return on S&P500 historical data of the greedy and simulated annealing methods for sparse mean-reverting portfolios of size L=3 and L=4

Next, I studied US swap rate data for maturities 1Y, 2Y, 3Y, 4Y, 5Y, 7Y, 10Y, and 30Y from 1995 to 2010. Similar to the S&P 500 analysis, I used sliding time windows of eight

observations to estimate model parameters, and applied the simple convergence trading methodology on heuristically selected mean reverting portfolios. Due to a poorer model fit, I observed more moderate returns of 15-45% over the 15 years for portfolio cardinalities of L=4-6. The methods generated best results for cardinalities L=4 and 5 (see Fig. 16)

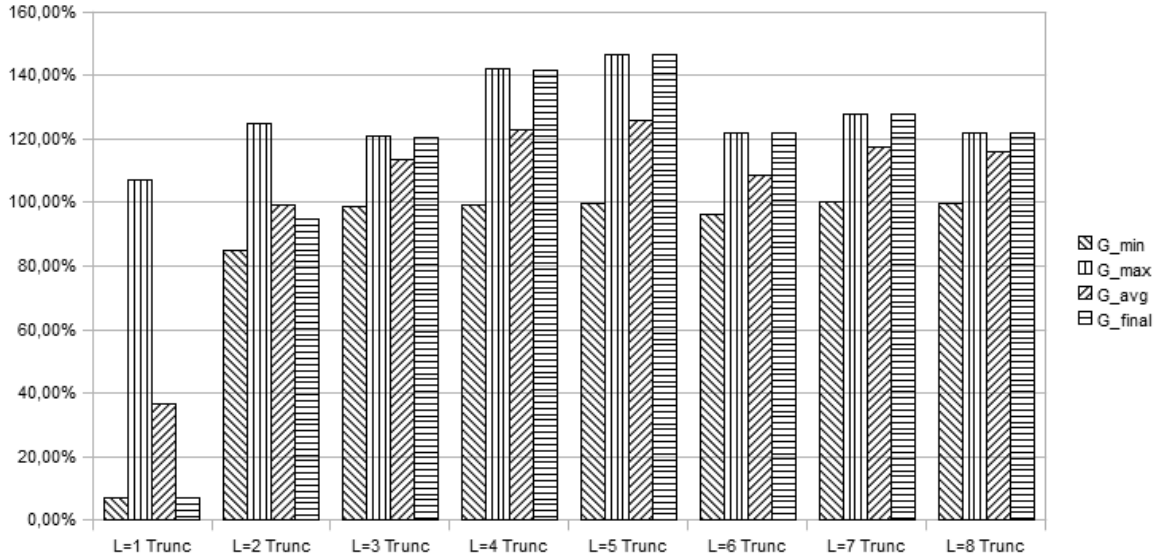

Fig. 16: Comparison of minimum, maximum, average and final return on historical US swap data of the truncation method for sparse mean-reverting portfolios of cardinality L=1-8

2.8. Conclusions and directions for future research

I have examined the problem of selecting optimal sparse mean-reverting portfolios based on observed and generated time series. After reviewing the novel parameter estimation techniques suggested in earlier publications, I have adapted simulated annealing to the problem of optimal portfolio selection. I have shown that the exhaustive method can be a viable practical alternative for problems with an asset universe size up to 25, and examined the relative performance of simulated annealing versus the greedy method whilst considering the theoretically best solution. I concluded that even for small problem sizes (ten assets and portfolios restricted to cardinality of five), simulated annealing can outperform the greedy method in 10% of the cases, very significantly in some cases. This ratio improved to 25% with the doubling of the problem size and cardinality restriction suggesting that simulated annealing becomes more appealing for larger problem sizes, whilst having the asymptotic runtime of simpler heuristics. Furthermore, the viability of the entire framework has been demonstrated by introducing a simple convergence trading strategy which proved to be very profitable on generated VAR(1) sequences and also viable on real historical time series of S&P 500 and US swap rate data. However, the fact that all four methods produced very similar profit profiles implies that the relationship between the mean reversion coefficient produced by the selected sparse portfolio and the profits achieved by trading is nontrivial. Our conclusion is that more factors (such as distance from the mean, model fit, etc.) needs to be taken into account in the selection of the objective function. However, the method can be directly applied to these more

complex objective functions, making the suggested numerical approach viable. Finally, more sophisticated trading methods, involving multiple portfolios and better cash management, could be developed to further enhance trading performance, and they could be analyzed with the introduction of transaction costs, short-selling costs or an order book.

CHAPTER 3

TASK SCHEDULING ON IDENTICAL MACHINES

3.1. Introduction

Scheduling algorithms were first developed in the fifties, considering applications in naval logistics. During the sixties, a significant amount of work was done on dynamic programming and integer programming formulations of scheduling problems [72]. During the seventies, computer scientists discovered scheduling as a tool for improving the performance of various computer systems [22] and started focusing on the computational complexity hierarchy of various algorithms. In the eighties, several research directions were pursued with increasing attention on stochastic scheduling algorithms [72]. In recent years, with the advent of grid and subsequently cloud computing, there is renewed interest in the field for many applications in computational biology, chemistry and finance [3].

Scheduling theory has a very important application in running large scale Monte Carlo simulations in financial services firms for evaluating risks and pricing. As a result, finding optimal schedules in real time which minimize the completion time of jobs subject to capacity constraints is an especially important task. More specifically, in the area of computational finance, the problems of portfolio selection, pricing and hedging of complex financial instruments requires an enormous amount of computational resources whose optimal usage is of utmost importance to financial services companies. The prices and risk sensitivity measures of complex portfolios need to be reevaluated daily, for which an overnight batch of calculations is scheduled and performed for millions of financial transactions, utilizing thousands of computing nodes. Each job has a well-defined priority and required completion time for availability of the resulting figures to the trading desk, risk managers and regulators. The jobs can generally be stopped and resumed at a later point on a different machine which is referred to as *preemption* in scheduling theory. For simplicity of modeling the problem, machines are generally assumed to be identical and there is a known, constant number of machines available. Tardiness of an individual job under a given schedule is defined as the amount of time by which the job finishes after its prescribed deadline, and is considered to be zero if the job finishes on or before the deadline.

This formalism actually quite closely resembles many practical applications in the financial sector. At many financial institutions, including Morgan Stanley, complex financial derivative transactions can only be evaluated using Monte Carlo pricing techniques. These calculations are performed on large-scale grid computing infrastructures where the nodes can be considered identical and jobs have well defined due dates and priorities. When evaluating the risk profile of large portfolios of such derivatives, computations are scheduled overnight on large "*compute farms*".

Even a 5-10% improvement in computation time via scheduling can result in figures being more readily available for traders, risk managers and regulators and can save millions of dollars in hardware resources. Therefore, this is an extensively studied problem with significant practical applications. As such, the problem of finding optimal schedules for jobs running on identical machines has been extensively studied over the last three decades. In his paper, Sahni presents an *O(n log mn)* algorithm to construct a *feasible* schedule, one that meets all due times, if one exists, for *n* jobs and *m* machines [77]. The basic idea of the algorithm is to schedule jobs with earliest due dates first, but fill up machines with smaller jobs if possible. Note that this method allows the development of an algorithm to compute the minimal amount of unit capacity (the capacity available at each time unit, assuming constant capacity throughout) for which a feasible schedule exists. This result has been extended to machines with identical functionality but different processing speed, termed *uniform machines,* and jobs with both starting times and deadlines; Martel constructs a polynomial time feasible schedule for this problem, if one exists [63]. However, the scheduling task becomes more difficult when a feasible schedule does not exist and the goal is to minimize some measure of delinquency, often termed *tardiness*.

In case of minimizing the maximum tardiness across all jobs, Lawler shows that the problem is solvable in polynomial time, even with some precedence constraints [53]. Martel also used his construction to create a polynomial time algorithm to find the schedule which minimizes maximum lateness[63]. However, if the measure concerns the total tardiness instead of the maximal one, then even the single machine, total tardiness problem (without weights) was proven to be NP-hard by Du and Leung [30]. A pseudopolynomial algorithm has been developed by Lawler for this problem, using dynamic programming, but this is for the 1-machine problem and does not have good practical runtime characteristics [52].

Once the NP-hardness of the TWT problem was established, most of the research work on the problem concerned the development of fast, heuristic algorithms. Dogramaci and Surkis propose a simple heuristic for the total (non-weighted) tardiness problem without preemption [31]. Rachamadugu and Morton then studied the identical machine, total weighted tardiness problem without preemption [74]. They proposed a myopic heuristic and compared this to earliest due date (EDD), weighted shortest processing time (WSPT) and Montagne's rule on small problem sizes (two or five jobs in total). Azizoglu and Kirca worked on an algorithm to find optimal schedule for the unweighted total tardiness problem without preemption, but their branch and bound exponential algorithm is too slow, in practice, for problems with more than 15 jobs [16]. Armentano and Yamashita examined the non-weighted problem without preemption [14], and starting from the KPM heuristic of Koulamas [50] improved upon it, using tabu search. Guinet applies simulated annealing to solve the problem with uniform and identical machines and a lower bound is presented in order to evaluate the performance of the proposed method [40]. More recently, Sen et al. surveyed the existing heuristic algorithms for the single-machine total tardiness and total weighted tardiness problems [79] while Biskup et al. [19] did this for the identical machines total tardiness problem and also proposed a new heuristic. Akyol and Bayhan [11] provide an excellent recent review of

artificial neural network based approaches to scheduling problems and proposes a coupled gradient network to solve the weighted earliness plus tardiness problem on multiple machines. The feasibility of the method is illustrated on a single 8-job scheduling problem. Maheswaran et al. developed a Hopfield Neural Network approach to the single machine TWT problem and showed that the results were encouraging for a specific 10-job problem [58].

During my research, I designed a novel polynomial time heuristic for the identical machine TWT problem, based on the Hopfield neural network approach which is shown to perform better than existing simple heuristics and has desirable scaling characteristics. First, I mapped the problem to a Quadratic Unconstrained Binary Optimization (QUBO) problem. Then, I reformulated this to be directly solvable by a Hopfield neural network and developed effective methods to iteratively set the heuristic parameters. Finally, I mapped the results back to admissible schedules and evaluated the performance against other benchmarks.

I have further improved this approach by intelligent selection of the starting point and considering perturbations to this starting point. I have tested the newly developed methods on a large number of randomly generated problems as well as on a real scheduling data set obtained from Morgan Stanley, one of the largest financial institutions in the world.

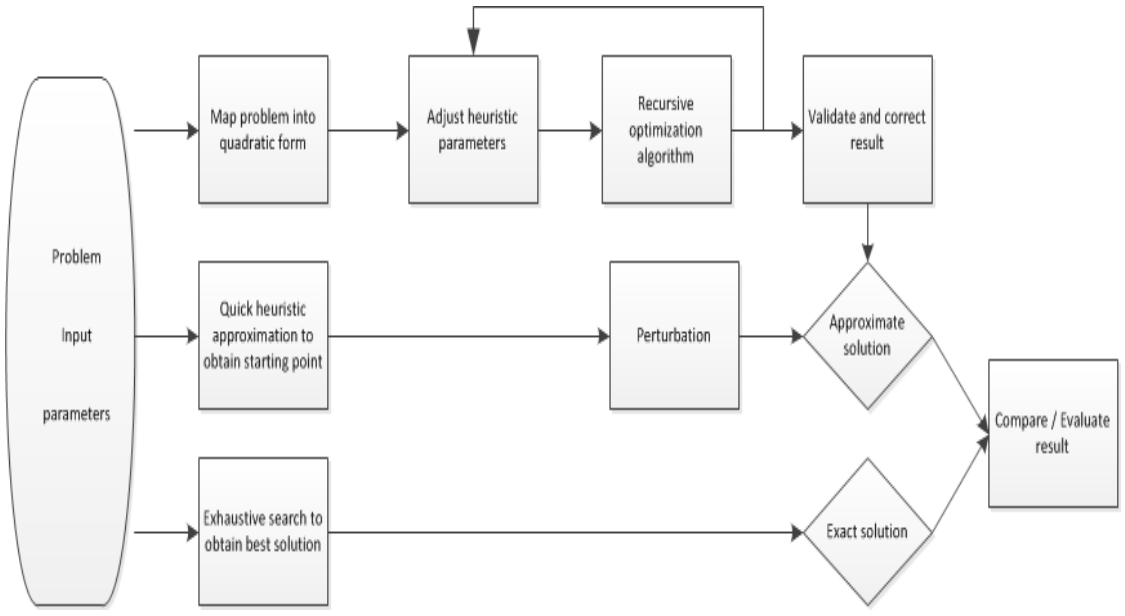

Fig. 17: Summary flow diagram of the heuristic approach to solving the total weighted tardiness scheduling problem.

3.2. Problem Formulation

In this section, I give a formal presentation for the problem of optimally scheduling jobs on finite number of identical processors under constraints on the completion times. The basic formalism is the following:

- Given N jobs with sizes $\mathbf{x} = \{x_1, x_2, x_3, \ldots x_N\} \in \mathbb{N}^N$. The processing of the jobs can be stopped and resumed arbitrary at any time, so the processing time units of each job need not be contiguous. In the literature this condition is known as *preemption* and also assumes a task started on one machine can continue on another [22].
- For each job a cutoff time is prescribed by $\mathbf{K} = \{K_1, K_2, K_3, \ldots, K_N\} \in \mathbb{N}^N$. This constraint defines the time within which the job is to be completed.
- The constant number of processors, the capacity of the system is denoted by $V \in \mathbb{N}$
- We are also given a vector $\mathbf{w} = \{w_1, w_2, w_3, \ldots, w_N\} \in \mathbb{R}^N$, $w_i \geq 0, \forall i = 1, ..., N$ denoting the relative priority (or weight) of each job, which we could use in the definition of the objective function.

A schedule is represented by a binary matrix $\mathbf{C} \in \{0,1\}^{N \times L}$ where $C_{i,j} = 1$ if job i is being processed at time slot j, and L denotes the length of the schedule. An example is given in (39) where the parameters are the following: $V = 2$, $N = 3$, $\mathbf{x} = \{2,3,1\}$, $\mathbf{K} = \{3,3,3\}$.

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{39}$$

The first row in (39) denotes the fact that under this schedule **C**, the first job is processed in time steps one and three (note that preemption is used as the processing of this job is not continuous) and therefore the prescribed size two of this job completes within the prescribed cutoff time of time step 3. Similarly, the three units of the second job complete within the cutoff time of three and the third job is completed ahead of the cutoff time on time step two. Summing the columns of matrix **C**, we see that the maximal capacity of V=2 is fully utilized on each time step.

In order to evaluate the effectiveness of a given schedule **C**, I define *tardiness* of a job as follows:

$$T_i = \max(0, F_i - K_i), \tag{40}$$

where F_i is the actual finish time of job i under schedule **C**: $F_i = \arg\max_j \{C_{i,j} = 1\}$ (The position of the last 1 in the i^{th} row in scheduling matrix $\mathbf{C}$.)

The problem which I want to solve can now be stated formally as follows:

$$\mathbf{C}_{opt} := \arg\min_{\mathbf{C}} \sum_{i=1}^{N} w_i T_i \tag{41}$$

Under the following constraints:

- The length of jobs in the scheduling matrix equals the pre-defined amounts

$$\sum_{j=1}^{L} C_{i,j} = x_i, \forall i = 1,...,N \tag{42}$$

- The scheduled jobs at a given time instant does not exceed the capacity of the system:

$$\sum_{i=1}^{N} C_{i.j} \leq V, \forall j = 1,...,L \tag{43}$$

I revisit the previous example with a minor change: $V = 2$, $N = 3$, $\mathbf{x} = \{2,3,2\}$ $\mathbf{K} = \{3,3,3\}$ and a weight vector $\mathbf{w} = \{3,2,1\}$. It can be observed that there is no solution, in which all jobs are completed before their cutoff times. A minimal weighted tardiness solution is the following:

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \tag{44}$$

3.3. Existing Heuristic Methods

Given the NP-hardness of the scheduling task as shown by Du and Leung [30], and therefore the amount of time it would take to find the exact, optimal solution, in most real-world settings the pragmatic approach of finding a fast, sub-optimal, but good solution is followed. As outlined in the introduction, this has been a very active field of research, and most studies use the Earliest Due Date (EDD) and Weighted Shortest Process Time (WSPT) heuristics as benchmarks for evaluating the proposed algorithms. In addition to these, I also outline the recently developed Load Balancing Scheduling (LBS) heuristic and a newly proposed Largest Weighted Process First (LWPF) heuristic. Furthermore, I also outline a random processing method which is used as a low benchmark for the testing results.

3.3.1. Earliest Due Date (EDD) Heuristic

The EDD heuristic orders the sequence of jobs to be executed from the job with the earliest due date to the job with the latest due date. Using the notation of chapter 2, I relabel the job indices so that the following inequality holds

$$K_1 \leq K_2 \leq ... \leq K_N \tag{45}$$

Once this ordering is determined, the jobs are allocated to the machines in this order, always utilizing the maximal capacity. Once a job finishes and capacity is freed up, the next job using the

above ordering is scheduled on the freed up machine. It has been shown in Pinedo 2008 that EDD finds the optimal schedule when one wants to minimize the maximum tardiness on a single machine. However, note that when the objective function includes the relative priorities of jobs, EDD is at a severe disadvantage, as it does not include consideration of the weights.

3.3.2. Weighted Shortest Processing Time (WSPT) Heuristic

The WSPT method is analogous to the EDD method in that it orders the jobs according to well-defined criteria and then schedules the jobs according to this ordering, utilizing the maximal capacity available. Once a job finishes and capacity is freed up, the next job using the ordering is scheduled. Using the notation of section 3.2, the jobs are ordered such that the below holds:

$$\frac{x_1}{w_1} \leq \frac{x_2}{w_2} \leq \ldots \leq \frac{x_N}{w_N} \tag{46}$$

3.3.3. Largest Weighted Process First (LWPF) Heuristic

This heuristic is analogous to the EDD and the WSPT heuristics with the jobs ordered according to the following inequality:

$$w_1 \leq w_2 \leq \ldots \leq w_N \tag{47}$$

This is a simple heuristic which allows us get a sense of the importance of considering the weights versus the cutoff times. I have not seen this simple heuristic mentioned anywhere in the literature, but it turns out to have quite good empirical characteristics which is one of the manz contributions of this paper.

3.3.4. Load Balancing Scheduling (LBS) Algorithm

Laszlo et al [51] suggest a novel deterministic, polynomial time algorithm for scheduling jobs with cutoff times, in the context of load balancing for wireless sensor networks. The basic idea of the algorithm is to start scheduling the jobs backwards from the maximal cutoff time, in order of decreasing cutoff times, always ensuring full capacity is utilized. I note that whilst the algorithm has proven to be extremely successful for load balancing, it does not take into account the weights of the jobs and therefore will suffer from the same shortcomings as EDD.

3.3.5. Random Method

In this method, the jobs are ordered randomly and are scheduled in such a way as to always fully utilize the maximal available capacity. This unsophisticated heuristic is useful as a low benchmark by repeating it a number of times and ensuring that other heuristics with more information outperform it.

3.4. Quadratic Programming and Hopfield Neural Network Based Approach for Scheduling

Since the number of binary matrices is exponential with the number of nodes multiplied by the length of the schedule, exhaustive search with complexity $\mathcal{O}\left(2^{N\cdot L}\right)$ is generally computationally infeasible to solve the optimization problem at hand. Thus, the goal is to develop a polynomial time approximate solution by mapping the problem to an analogous quadratic optimization problem, similar to the approach of Levendovszky et al. [54] and Treplan et al. [84]. The following figure shows the overall flow diagram of my approach

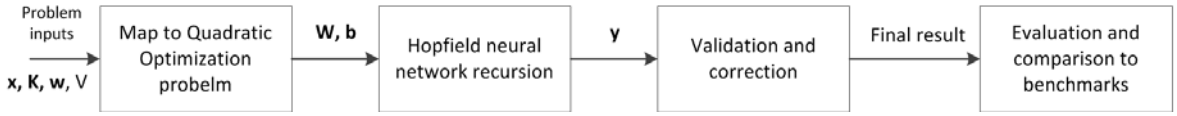

Fig. 18: Overall flow diagram of my novel HNN approach to solving scheduling problems

I first review the methods available for optimizing equations in quadratic form. Let us assume that matrix $\mathbf{W}$ is a symmetric matrix of type $n\times n$ and vector $\mathbf{b}$ is of length n. I seek the optimal n dimensional vector $\mathbf{y}$ which minimizes the following quadratic function [68]:

$$f\left(\mathbf{y}\right)=-\frac{1}{2}\mathbf{y}^{T}\mathbf{W}\mathbf{y}+\mathbf{b}^{T}\mathbf{y}, \tag{48}$$

subject to one or more constraints of the form of

$$\begin{aligned}\mathbf{A}\mathbf{y}&\leq\mathbf{v}\\ \mathbf{B}\mathbf{y}&=\mathbf{u}\end{aligned} \tag{49}$$

If the problem at hand contains only linear constraints then it can be solved as presented by Murty [65] in the linear case and as presented by Levendovszky et al [54] in the discrete binary case. In the non-linear constraints case, if the matrix $\mathbf{W}$ is positive definite, then the function $f(\mathbf{y})$ is convex and the problem can be solved with the ellipsoid method presented by Zhi-quan [86]. When $\mathbf{W}$ is indefinite the problem is NP-hard [76].

A frequently used powerful heuristic algorithm to solve quadratic optimization problems is the Hopfield Neural Network (HNN). This neural network is described by the following state transition rule:

$$\mathbf{y}_{i}(k+1)=\operatorname{sgn}\left(\sum_{j=1}^{N}\hat{W}_{ij}y_{j}(k)-\hat{b}_{i}\right),i=\operatorname{mod}_{N}k, \tag{50}$$

where

$$\begin{aligned}\mathbf{d} &= -\mathrm{diag}(\mathbf{W}) \\ \widehat{\mathbf{W}} &= -\mathbf{W} - \mathrm{diag}(\mathbf{d}) \\ \hat{\mathbf{b}} &= \mathbf{b} - \frac{1}{2}\mathbf{d}.\end{aligned} \tag{51}$$

Note that $i = \mathrm{mod}_N k$ implies asynchronous, sequential updates to the elements of the $\mathbf{y}$ vector. There exist techniques for synchronous updating, but these require hardware-level implementation and have different convergence properties. For the purposes of this dissertation, we will focus on the asynchronous, CPU-based implementation and leave investigation of the other methods as an area for future research.

Using the Lyapunov method, Hopfield proved that HNN converges to its fix point, as a consequence HNN minimizes a quadratic Lyapunov function [42]:

$$\mathcal{L}(\mathbf{y}) := -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\hat{W}_{ij}y_i y_j + \sum_{i=1}^{N} y_i \hat{b}_i = -\frac{1}{2}\mathbf{y}^T\hat{\mathbf{W}}\mathbf{y} + \hat{\mathbf{b}}^T\mathbf{y}. \tag{52}$$

Thus, HNN is able to solve combinatorial optimization problems in polynomial time under special conditions as it has been demonstrated by Mandziuk [59] and Mandziuk and Macukow [60].

These methods motivate us to map the existing optimization problem into quadratic form. First the binary scheduling matrix $\mathbf{C}$ is mapped into a binary column vector $\mathbf{y}$ as follows:

$$\mathbf{C} = \begin{pmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,L} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,L} \\ \vdots & \vdots & \ddots & \vdots \\ C_{N,1} & C_{N,2} & \cdots & C_{N,L} \end{pmatrix} \rightarrow \tag{53}$$

$$\mathbf{y} = \left(C_{1,1}, C_{1,2}, \cdots, C_{1,L}, C_{2,1}, \cdots, C_{2,L}, C_{N,1}, \cdots, C_{N,L}\right)^T. \tag{54}$$

The original objective function (41) is extended as follows:

$$\min \sum_{i=1}^{N} w_i \left(\max\left(0, \arg\max_j \left\{C_{i,j} = 1\right\} - K_i\right)\right) \tag{55}$$

This objective is equivalent to:

$$\min \sum_{i=1}^{N} w_i \left(\sum_{j=K_i+1}^{L} C_{i,j} \right) \tag{56}$$

The minimization problem is thus equivalent to:

$$\mathbf{C}_{opt} := \arg\min_{\mathbf{C}} \sum_{i=1}^{N}\sum_{j=K_i+1}^{L} w_i {C^2}_{i,j} \tag{57}$$

Therefore, my task was to find matrix $\mathbf{W}_A$ and vector $\mathbf{b}_A$ to satisfy the below:

$$-\frac{1}{2}\mathbf{y}^T\mathbf{W}_A\mathbf{y} + {\mathbf{b}_A}^T\mathbf{y} = \sum_{i=1}^{N}\sum_{j=K_i+1}^{L} w_i {C^2}_{i,j} \tag{58}$$

Using the definition of **y** from (54) then expanding and rearranging the right side of (58) into vector and matrix format, I found the below solution:

$$\mathbf{b}_A = \mathbf{0}_{JL\times 1}, \tag{59}$$

$$\mathbf{W}_A = -2\begin{pmatrix} \mathbf{D}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{D}_J \end{pmatrix} \tag{60}$$

where

$$\mathbf{D}_j = \begin{pmatrix} \mathbf{0}_{K_j\times K_j} & \mathbf{0}_{K_j\times L-K_j} \\ \mathbf{0}_{L-K_j\times K_j} & w_j * \mathbf{I}_{L-K_j\times L-K_j} \end{pmatrix} \in \mathbb{R}^{L\times L} \tag{61}$$

Having transformed the objective function to a quadratic form, I now turn my attention to doing the same with the two constraints outlined in (42) and (43). In this way, the constraints become part of the objective function with appropriate heuristic coefficients.

The constraints in (42) can be rewritten as follows:

$$\forall i : \sum_{j=1}^{L} C_{i,j} = x_i \rightarrow \min \sum_{i=1}^{N}\left(\left(\sum_{j=1}^{L} C_{i,j}\right) - x_i\right)^2 \tag{62}$$

This will lead to the following equation:

$$-\frac{1}{2}\mathbf{y}^T\mathbf{W}_B\mathbf{y} + \mathbf{b}_B{}^T\mathbf{y} = \sum_{i=1}^{N}\left(\left(\sum_{j=1}^{L} C_{i,j}\right) - x_i\right)^2 \tag{63}$$

The solution of equation (63) is:

$$\mathbf{b}_B = 2\begin{pmatrix} x_{1_{1\times L}} & x_{2_{1\times L}} & \cdots & x_{J_{1\times L}} \end{pmatrix} \tag{64}$$

$$\mathbf{W}_B = -2\begin{pmatrix} \mathbf{1}_{L\times L} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{L\times L} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1}_{L\times L} \end{pmatrix} \tag{65}$$

Another constraint ensures that the scheduling does not overload the processing unit, described in (43). The required transformation is as follows:

$$\forall j : \sum_{i=1}^{N} C_{i.j} = V \rightarrow \min \sum_{i=1}^{L}\left(\left(\sum_{j=1}^{N} C_{i,j}\right) - V\right)^2 \tag{66}$$

Please note the technicality that this constraint may not be relevant for the last few columns where only the remaining jobs have to be scheduled and schedule matrices not exhausting the full capacity should not be penalized. The total length of scheduling can be written as follows:

$$\tilde{L} = \left\lceil \frac{\sum_{i=1}^{N} x_i}{V} \right\rceil$$
$$L = \max\left(\tilde{L}, \max_i x_i, \max_i K_i\right) \tag{67}$$

The number of columns where capacity V needs to be fully utilized is the following:

$$M = \max\left(1, \max_i x_i - L + 1\right) \tag{68}$$

Taking into consideration (68) the mapping of (66) can be described with the following equation:

$$-\frac{1}{2}\mathbf{y}^T\mathbf{W}_C\mathbf{y} + \mathbf{b}_C{}^T\mathbf{y} = \sum_{i=1}^{M}\left(\left(\sum_{j=1}^{N} C_{i,j}\right) - V\right)^2 \tag{69}$$

The solution of this equation is the following:

$$\mathbf{b}_C = \left[\mathbf{V}_{M\times 1}, \mathbf{0}_{L-M\times 1}, \mathbf{V}_{M\times 1}, \mathbf{0}_{L-M\times 1}, \ldots, \mathbf{V}_{M\times 1}, \mathbf{0}_{L-M\times 1}\right], \tag{70}$$

$$\mathbf{W}_C = -2\begin{pmatrix} \mathbf{D} & \mathbf{D} & \cdots & \mathbf{D} \\ \mathbf{D} & \mathbf{D} & \cdots & \mathbf{D} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{D} & \mathbf{D} & \cdots & \mathbf{D} \end{pmatrix}, \tag{71}$$

where

$$\mathbf{D} = \begin{pmatrix} \mathbf{I}_{M\times M} & \mathbf{0}_{M\times L-M} \\ \mathbf{0}_{L-M\times M} & \mathbf{0}_{L-M\times L-M} \end{pmatrix}. \tag{72}$$

I can combine these mappings into the form of equation (48) as follows:

$$\mathbf{W} = \alpha\mathbf{W}_A + \beta\mathbf{W}_B + \gamma\mathbf{W}_C \in \mathbb{R}^{NL\times NL} \tag{73}$$

and

$$\mathbf{b} = \alpha\mathbf{b}_A + \beta\mathbf{b}_B + \gamma\mathbf{b}_C \in \mathbb{R}^{NL\times 1} \tag{74}$$

Note that, the objectives can be controlled with heuristic coefficients α, β, γ in order to strike a good tradeoff between the weights of different requirements. Having this quadratic form at hand, I can apply the HNN to provide an approximate solution to the optimization problem in polynomial time.

3.5. Improvements to HNN and other heuristics

In this section, three algorithms are proposed in order to improve upon the existing methods. Unfortunately, traditional algorithms which provide fast convergence often fail to reach the global

optimum. As such, the development of these novel heuristics is motivated by the desire to avoid getting stuck in local minima and still reach good quality solutions in real time by exploiting the fast convergence of the HNN. The proposed methods are based on the careful choice of the initial state or random perturbation of the initial state.

3.5.1. Smart Hopfield Neural Network (SHNN)

The SHNN approach uses the result of the LFPW heuristic as input for the HNN recursion. The block diagram of the method is provided in Fig. 19. In this way the HNN recursion is not being repeated several times with different starting points, therefore the method can be executed faster than the original HNN method; however repetitions due to the tuning of the heuristic parameters are still required.

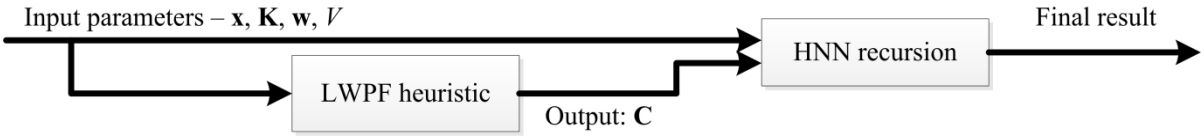

Fig. 19: Block diagram of the SHNN method

The intuition behind this method is that since the value of the Lyapunov function monotonically decreases during the HNN recursion, a better starting point may converge faster and, depending on the shape of the Lyapunov surface to be minimized, could converge to a better quality minimum than a random starting point.

3.5.2. Perturbed Smart Hopfield Neural Network (PSHNN)

The SHNN method is faster than the original HNN solution, as it only considers a single starting point. However, as demonstrated by numerical simulations, it can get stuck in local minima. This can be addressed by considering random perturbations to the "smartly" selected initial starting point. The result of the LFPW method is perturbed randomly, in order to generate a set of initial points for the HNN recursion. The block diagram of this method is shown in Fig. 20.

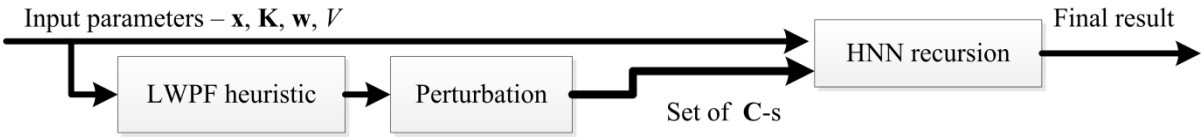

Fig. 20: Block diagram of the PSHNN method

Fig. 21 provides a visual explanation as to why perturbing the initial point may have a beneficial effect when the HNN iteration gets stuck in a local minimum.

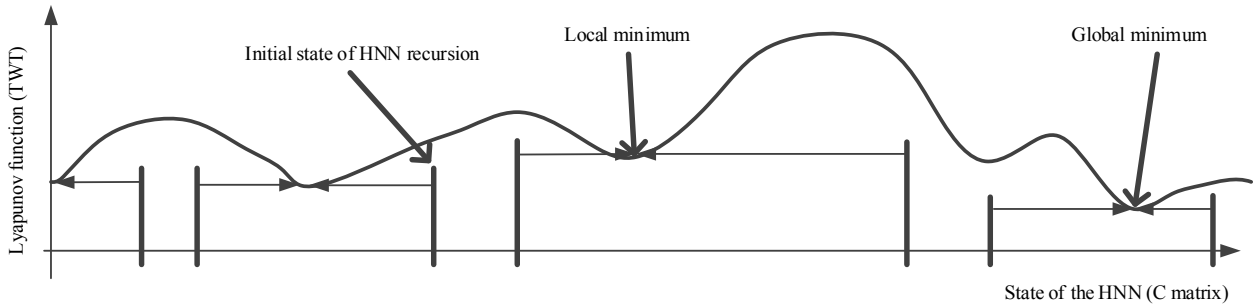

Fig. 21: Illustration of the advantage of multiple perturbed starting points in finding the global minimum

The algorithm which generates the set of inputs is described by Algorithm 3, where the function $\text{random}(\Theta)$ produces a uniformly distributed random integer value in range Θ and the function $\text{correct}(\mathbf{C},\mathbf{X})$ is an error correction function, given by Algorithm 4.

The degree of perturbation is an input parameter which in the implementation is proportional to the number of the columns of scheduling matrix. The final result is the best solution provided by the HNN recursions from the set of input points generated by the algorithm.

perturb($\mathbf{C}_{LWPF}$, $N, D, \mathbf{x}$ **) -** (Initial **C**, Number of outputs, Degree of perturbation, Job sizes)

$C_{RES} = \varnothing;$

FOR $i = 1 \rightarrow N$

$\mathbf{C} = \mathbf{C}_{LWPF}$

FOR $j = 1 \rightarrow D$

$\{k_1, k_2, l\} \leftarrow \{\text{random}([1..J]),\ \text{random}([1..J]),\ \text{random}([1..L])\} \wedge \{\mathbf{C}_{k_1,l}, \mathbf{C}_{k_2,l}\} \leftarrow \{\mathbf{C}_{k_2,l}, \mathbf{C}_{k_1,l}\}$

END FOR

$\mathbf{C} = \text{correct}(\mathbf{C}, \mathbf{x})$

$C_{RES} = \{C_{RES}, \mathbf{C}\}$

END FOR

RETURN C_{RES}

Algorithm 3. Aglorithm for randomly perturbing scheduling matrices

3.5.3. Perturbed Largest Weighted Process First (PLWPF)

Analogously to the relationship between the SHNN and PSHNN methods, the PLWPF is the perturbed version of the LWPF method. The block diagram is shown in Fig. 22.

Fig. 22: Block diagram of the PLWPF method

The random perturbation is performed using Algorithm 3, and among the perturbed **C** matrices, the one which produces the lowest value for the objective function is chosen as the final result of the PLWPF method.

3.6. Numerical Results

In this section, the performance of the novel heuristic approaches is investigated and is compared to the performance of already known heuristics.

3.6.1. Simulation method

Each method outlined above has been tested by extensive simulations on a large and diverse set of input parameters with the aim to characterize the algorithms empirically on scheduling problems of different sizes. The algorithms were implemented in MATLAB and tests were run in this simulation environment with randomly generated parameters such as size of jobs, cut-off times, and weights. The size of each job and its cut-off time and corresponding weight are generated as follows:

$$\mathbf{x}_i = \text{random}\left(\left[1, c_1\right]\right), \tag{75}$$

$$\mathbf{K}_i = \mathbf{x}_i + \text{random}\left(\left[c_1, 1.5 \cdot c_1\right]\right), \tag{76}$$

$$\mathbf{w}_i = \text{random}\left(\left[1, c_2\right]\right), \tag{77}$$

where $\text{random}(\Theta)$ produces a uniformly distributed random integer value in range Θ.

In the simulations the constant c_1 was set to 10 and c_2 was set to 5. The number of processors (V) was determined as follows:

$$V = 0.25 \cdot J. \tag{78}$$

The problem is expected to be solved without tardiness when the capacity is such that V=J. Therefore (78) ensures that there is a high likelihood of tardiness associated with the generated problem. For each problem size (dimension of job vector), 100 different problems were generated randomly, using (75)-(78) and the results of the methods were compared in each case.

The result of LWPF was used as the initial state for SHNN. Also, this result was perturbed randomly and these perturbations were used as starting points for PSHNN. The best perturbed result is the result of PLWPF.

The HNN and PSHNN methods were repeated 1000 times for each problem with different random or perturbed starting points and the best solution was used. In addition, the heuristic parameters α, β and γ were adjusted between the simulations in order to provide a good balance between optimizing the objective function, but also meeting the required constraints to produce a valid scheduling matrix. To adjust the heuristic parameters I used Algorithm 4.

REQUIRE $\mathbf{x}, \mathbf{K}, V, e$

$\alpha \leftarrow 0.1, \beta \leftarrow 5, \gamma \leftarrow 5$

$i \leftarrow 0$

REPEAT

$i \leftarrow i + 1$

$\mathbf{C}_i \leftarrow \mathrm{HNN}(\mathbf{x}, \mathbf{K}, V, \alpha, \beta, \gamma)$

$\alpha \leftarrow \alpha + 0.01$

UNTIL $\mathrm{error}(\mathbf{C}_i) \leq e$

FOR $k = 1 \rightarrow i$

$\mathbf{C}_k \leftarrow \mathrm{correct}(\mathbf{C}_k)$

$\mathbf{T}_k \leftarrow \mathrm{calculateTWT}(\mathbf{C}_k)$

END FOR

RETURN $\min(\mathbf{T})$

Algorithm 4. Algorithm for adjusting the heuristic parameters

In order to achieve valid solutions in all cases by HNN, I introduced an error correction function (see Algorithm 5) in order to cut and replace the unnecessary 1-s in the scheduling matrix. In the simulations, parameter *e* was set to 5. All of the results presented in the following sections therefore concern valid schedules meeting the required constraints.

Correct($\mathbf{x}, \mathbf{w}, \mathbf{K}, V, \mathbf{C}$ **)**

FOR $k = 1 \rightarrow L$

WHILE $\sum_{i=1}^{N} \mathbf{C}_{i,k} > V$

Remove 1 from row *j*, where *j* is the row in column *k* with minimal weight that has $C_{j,k} = 1$

END WHILE

END FOR

FOR $k = 1 \rightarrow N$

WHILE $\sum_{i=1}^{L} \mathbf{C}_{k,i} > \mathbf{x}_k$

Remove 1 from row *k* from the column *l*, where *l* is the righternmost column in row *k* such that $C_{k,l} = 1$

END WHILE

WHILE $\sum_{i=1}^{L} \mathbf{C}_{k,i} < \mathbf{x}_k$

Add 1 to row *k* in column *l*, where *l* is the lefternmost column where 1 can be added without violating the capacity constraint *V*

END WHILE

END FOR

RETURN C

Algorithm 5. Correction algorithm for the scheduling matrix produced by the HNN

3.6.2. Comparison of HNN performance versus other known heuristics

The first simulation compares the average total weighted tardiness provided by the algorithms for different problem sizes. Fig. 23 demonstrates that the best solution achieved by the HNN produces better average TWT for all problem sizes than any of the other heuristics. The performances of the EDD, LBS and random methods are, on average, worse than the WSPT and LWPF heuristics and the HNN for all problem sizes. This shows that consideration of the weights in the optimization problem is critical to reach near-optimal TWT schedules. It is also clearly visible that HNN consistently and significantly outperforms all other methods for all problem sizes. Note also that the simple LWPF heuristic, proposed by us, outperforms the WSPT heuristic widely used as a benchmark in the literature.

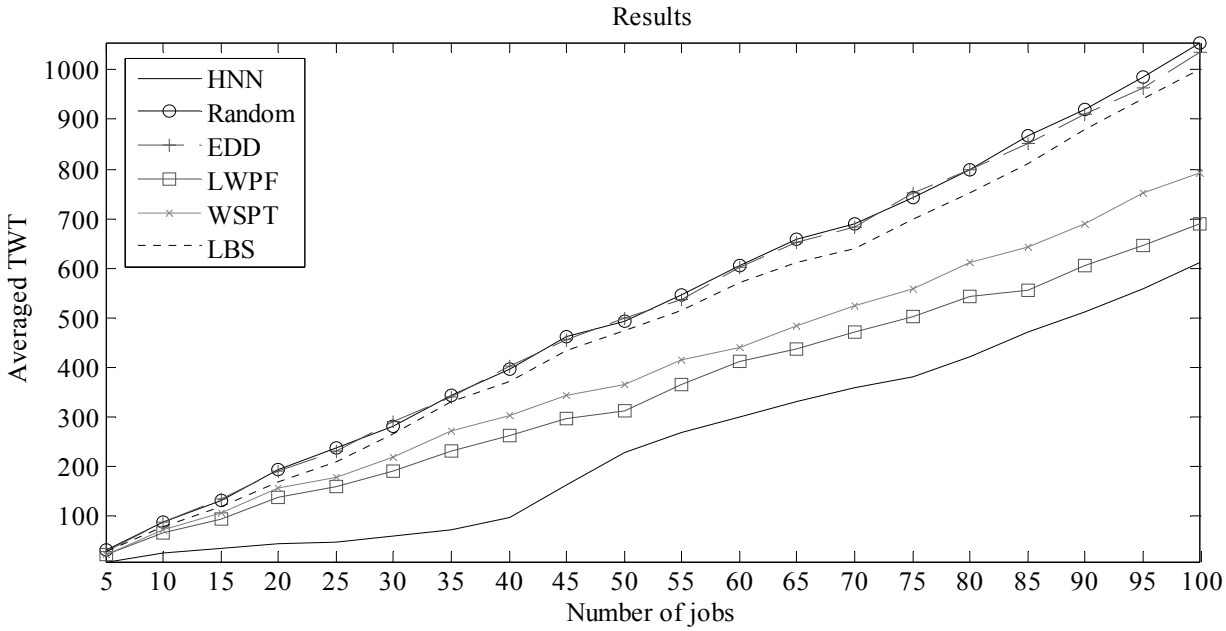

Fig. 23: Average TWT produced by each heuristic over randomly generated problems depicted as a function of the number of jobs in the problem.

In order to gain a better understanding of HNN's relative outperformance of other heuristics, I also computed the relative difference in TWT produced by HNN versus each of the other heuristics. The results are depicted in Fig. 24.

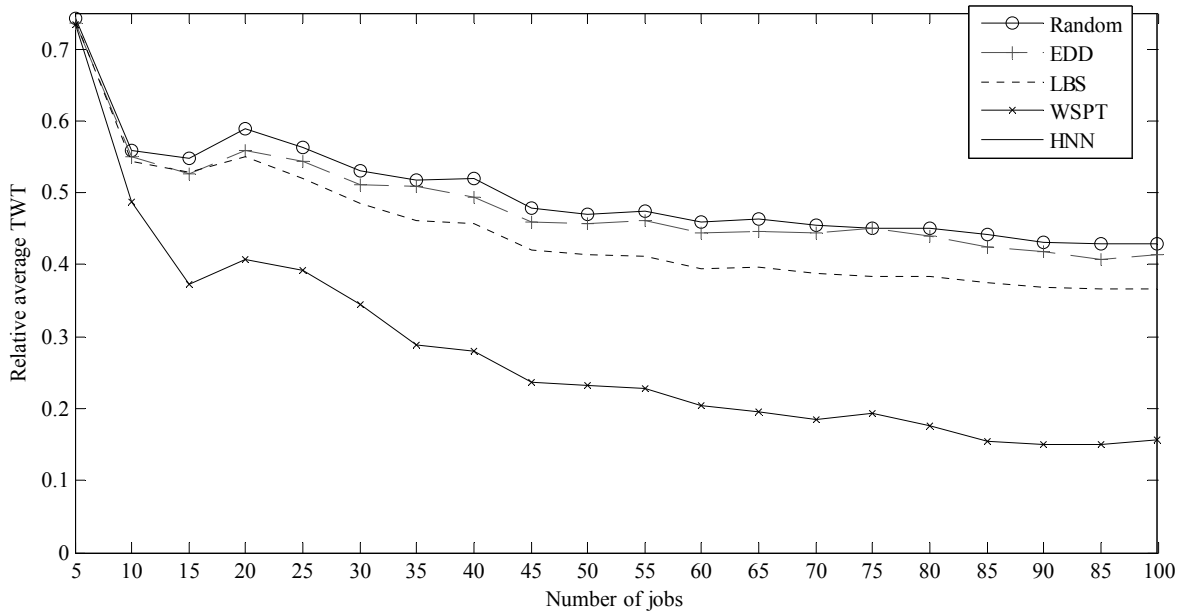

Fig. 24: Relative average TWT gain produced by HNN versus the other heuristics over randomly generated problems as a function of the number of jobs in the problem.

Fig. 25 depicts the ratio of average TWT produced by the HNN method versus the other heuristics. I conclude that, on average, the best solution of the HNN heavily outperforms the traditional solutions: the total weighted tardiness of the best HNN solution is 30%-65% of the EDD, 25%-85% of the LWPF, and 40%-90% of the WSPT. However, as the problem size increases, the

WSPT and LWPF heuristics produce solutions which are closer to the HNN which is consistent with what we saw on Fig.23 also.

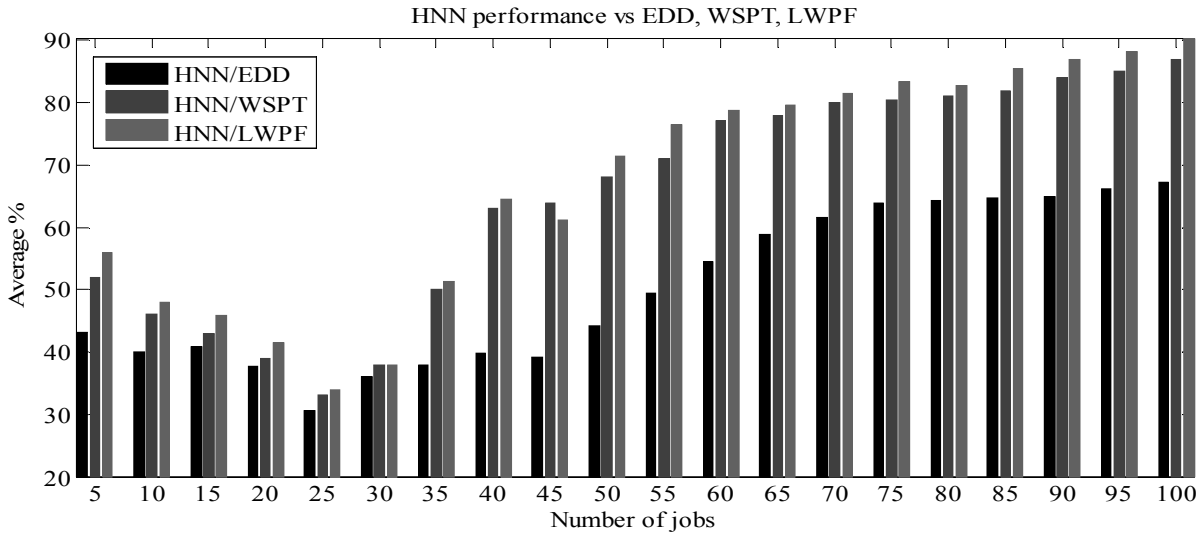

Fig. 25: Ratio of average TWT produced by HNN versus EDD, WSPT and LWPF as a funciton of the problem size

In order to verify that the HNN consistently outperforms the other heuristics on a wide spectrum of problems, not just on a few selected problems for each given problem size, I ran a more detailed experiment. In this case, I investigated 500 randomly generated problems for each problem size and computed the percentage of times HNN produced a better solution than LWPF. Table 3 shows the result which quite convincingly proves the general applicability of HNN to different problem types. Across the spectrum of problem domain, the ratio is higher than 98.5 % for all investigated job sizes.

Job size	**5**	**10**	**15**	**20**	**30**	**40**	**50**	**75**	**100**
	99,9%	100%	100%	99,5%	99,2%	99,6%	99,3%	98,6%	98,8%

Table 3. Percentage of problems in which HNN provides an improved solution over the next best heuristic, LWPF

3.6.3. Analysis of the performance of the improved HNN methods

The first simulation compares the average total weighted tardiness provided by the algorithms for different problem sizes. For each problem size, 100 different problems were generated randomly, using (75)-(78) and the results of the methods were compared to the theoretically best schedule, obtained by exhaustive search in Fig. 26 up to a problem size of 35 jobs. Simulations then were extended to problem sizes of up to 100 jobs and the difference of the resulting average TWT of each heuristic method to the result of the PSHNN method is depicted in Fig. 27. It can be observed that the best solution achieved by the PSHNN method has better average

TWT for all problem sizes than any of the other heuristics. Previously, I showed that the performance of the LWPF method is, on average, worse than the HNN method for all problem sizes [3]. However, I see that random perturbation does improve LWPF, as the solution achieved by PLWPF method is better than the solution of the original LWPF method for all problem sizes. The SHNN solution can be evaluated faster than HNN; however, its average TWT is only slightly better than the HNN method in case of small number of jobs and is significantly worse for larger problem sizes. Therefore, SHNN as a method in itself is not that impressive; however, as a stepping stone towards the PSHNN method its role proved to be very important.

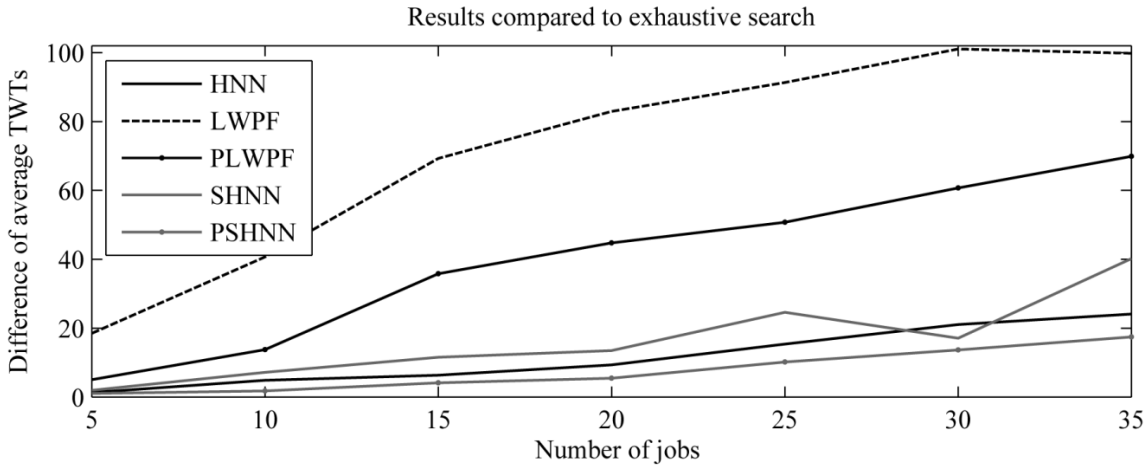

Fig. 26: The difference between the average TWT produced by each heuristic and the theoretical best schedule obtained by exhaustive search, over 100 randomly generated problems for each problem size.

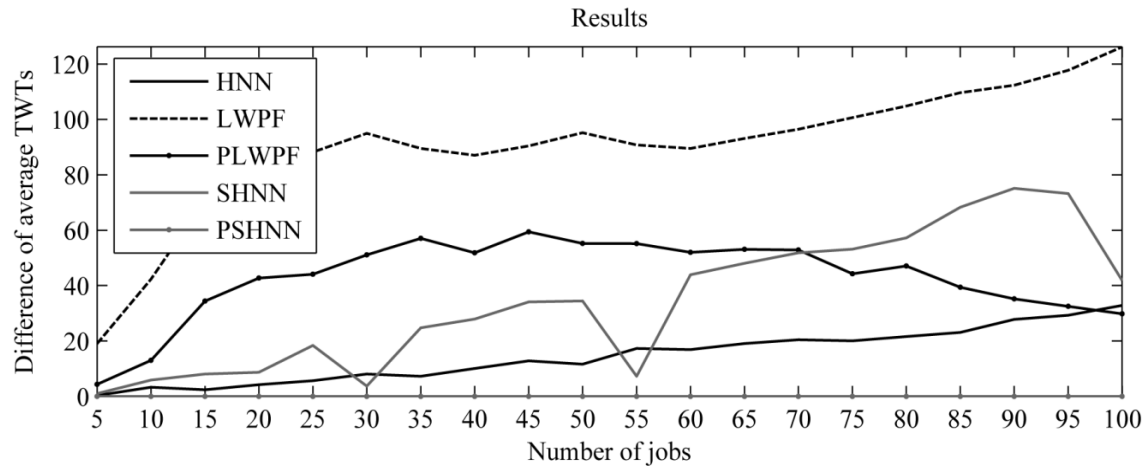

Fig. 27: The difference between the average TWT produced by each heuristic and the average TWT produced by the best method, PSHNN over 100 randomly generated problems for each problem size.

I conclude that, on average, the best solution of the PSHNN outperforms the traditional solutions: the total weighted tardiness of the best PSHNN solution is 93%-97% of the HNN, 68%-96% of the LWPF; the TWT of the best PLWPF solution is 47%-95% of the original LWPF; the TWT of the SHNN solution is 77%-110% of the PLWPF. As the problem size increases, the PLWPF heuristic produce solutions which are closer to the PSHNN. The ratio between the HNN and PSHNN is relatively stable independently of the problem size.

In order to verify that the PSHNN consistently outperforms the other heuristics on a wide spectrum of problems, not just on a few selected problems for each given problem size, I ran a more detailed numerical experiment. In this case, I investigated 500 randomly generated problems for each problem size and computed the percentage of times the PSHNN produced a better solution than HNN, the next best method. Table 4 shows the result which quite convincingly proves the general applicability of the PSHNN to different problem types. Across the spectrum of problem domain, the ratio is higher than 99.2% for all investigated job sizes.

Job size	**5**	**10**	**15**	**20**	**30**	**40**	**50**	**75**	**100**
	100%	100%	99,9%	99,8%	99,8%	99,9%	99,7%	99,7%	99,5%

Table 4. Percentage of problems in which PSHNN provides an improved solution over the next best heuristic, HNN.

3.6.4. Parallel implementation of the algorithms

In this section, for further speed-up a possible implementation of the HNN on GPGPU, specifically on Fermi architecture [47] is presented. Table 5 contains the theoretical order of convergence of the algorithms as a function of the length of the input parameters.

Algorithm	**Theoretical order of convergence**
LWPF	$\mathcal{O}(L \cdot N)$
HNN	$\mathcal{O}(L^2 \cdot N^2)$ (Hopfield 1982, Haykin 2008)

Table 5. Theoretical runtime of the algorithms

Although the number of repetitive iterations does not affect the theoretical order of convergence, the actual runtime of the algorithms may increase significantly. In order to reduce overall runtimes, each HNN run within the PSHNN method could be parallelized, resulting in multiple HNN runs from perturbed starting points taking the same time as a single HNN run. Furthermore, Liang [56] introduced a highly parallelized implementation of the HNN, which allows for significant speedup of each HNN run. More formally, the total runtime T of the algorithm is given by:

$$T = R_{init}\left(R_{heur}\,t\right) \tag{79}$$

Where R_{init} denotes the number of repeats of the algorithms from different initial starting points, R_{heur} denotes the number of repeats of the algorithm with different heuristic parameter (alpha, beta, gamma) settings and t denotes the runtime of a single non-parallelized HNN run. For example, assuming a 200×200 sized **W** weight matrix corresponding to 30 jobs to be scheduled, with R_{init} = 500, R_{heur} =100, the estimated time of execution on a 3GHz CPU processor is 50 seconds running optimized MATLAB code.

Since each iteration is independent of other iterations, it can be executed independently and in parallel. Using an existing parallel HNN implementation, the parallelism on nVidia Fermi architecture is extended as follows [34]:

- The **W** matrix is sparse (5-10 % of values are nonzero, and there are patterns amongst the nonzero values), therefore in case of large **W** matrix one HNN iteration can be executed by a stream multiprocessor (SM) using its local (shared) memory.
- The independent HNN iterations can be executed on separate SM-s.

As such, the execution time of the PSHNN algorithm can be significantly decreased on Fermi multicore architecture, thus I can achieve a better solution than the other heuristics within the same execution time. The idea of the parallelization is illustrated in Fig. 28 and the execution times are shown in Fig. 29.

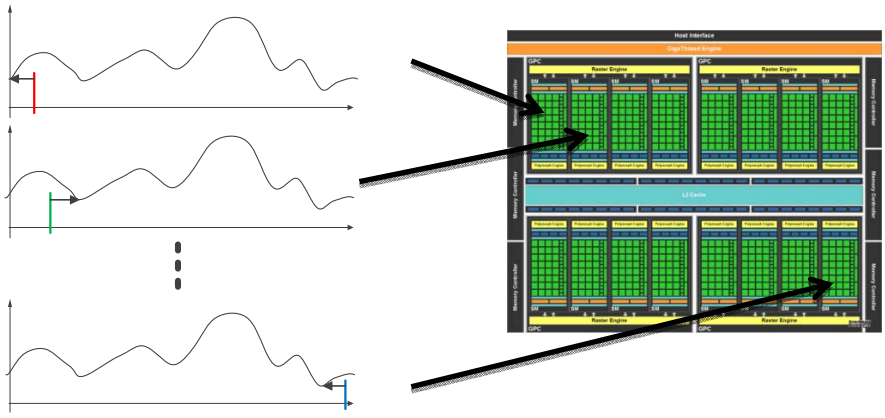

Fig. 28: Illustration of how each run of the perturbed starting point may be parallelized in a GPGPU infrastructure.

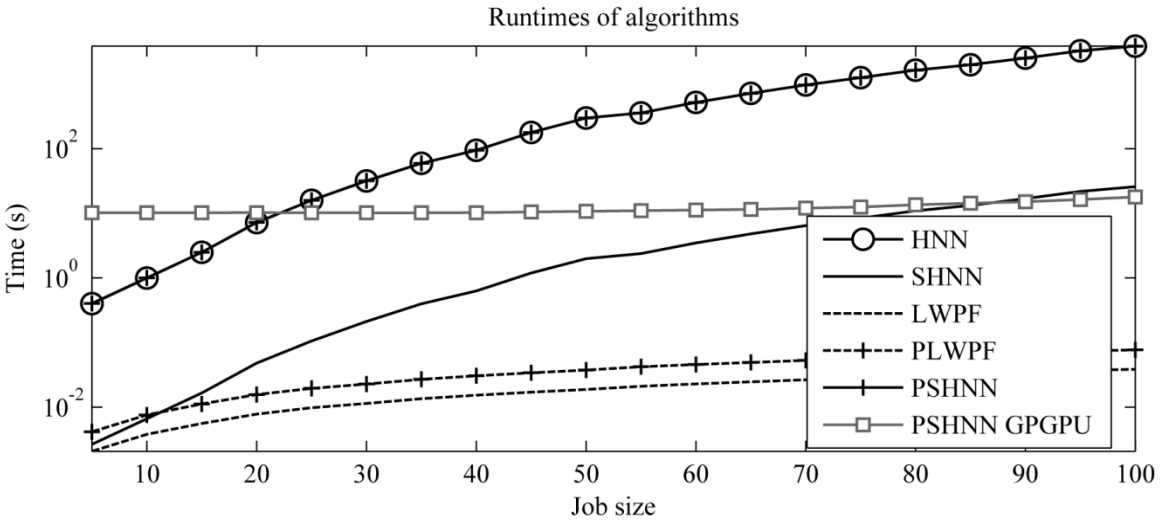

Fig. 29: Runtimes of the different algorithms, as a function of job size.

3.6.5. Practical case study on large problem size

A common task arising in computational finance is the scheduling of a set of valuation and risk sensitivity calculations which need to be run overnight based on the close-of-market market observables. Complex derivative transactions cannot, in general, be priced analytically and require Monte Carlo simulations for their valuation. In the specific example studied here, I consider the risk calculations associated with nearly 100 different portfolios, yielding altogether 556 jobs of an average size of 792 seconds to be scheduled. Each job has associated with it a cutoff time, by which time risk calculations need to finish in order for the results to be incorporated in the nightly firmwide Value at Risk (VaR) calculation and in order to have the sensitivity figures available for the trading desk at market open. Furthermore, each job has a priority associated with it which is assigned based on the relative amount of risk carried in that portfolio. I obtained specific runtime, deadline and priority data for an actual overnigh scheduling problem for traded portfolios at Morgan Stanley, one of the largest financial institutions in the world.

Having run each of the algorithms on this set of real data, Table 6 summarizes the results. As seen, PSHNN outperforms the next best method, PLWPF by 4.7% in terms of TWT, HNN by 7%, LWPF by 10% and EDD by a striking 48%. Another way to interpret the results is that PSHNN essentially completes all jobs with top priorities (8, 9, 10) on time, whilst PLWPF has a total tardiness of around 53 minutes (four average sized jobs) on these top priority jobs although it completes the less important jobs more quickly.

Total Weighted Tardiness

Weights	3	4	5	6	7	8	9	10	SUM	Increment to PSHNN
PSHNN	4,401	11,116	4,020	1,620	1,092	8	0	0	22,257	0 %
PLWPF	3,513	9,624	5,130	1,788	490	312	2,304	190	23,351	5 %
HNN	4,404	11,040	4,735	1,824	1,092	456	468	0	24,019	7 %
LWPF	4,404	11,140	5,470	2,472	1,183	40	0	0	24,709	10 %
EDD	4,401	9,940	1,770	636	1,134	464	22,752	1,430	42,527	48 %

Total Tardiness (Minutes)

Priority	3	4	5	6	7	8	9	10	SUM
PSHNN	244.5	463.2	134.0	45.0	26.0	0.2	0.0	0.0	912.8
PLWPF	195.2	401.0	171.0	49.7	11.7	6.5	42.7	3.2	880.8
HNN	244.7	460.0	157.8	50.7	26.0	9.5	8.7	0.0	957.3
LWPF	244.7	464.2	182.3	68.7	28.2	0.8	0.0	0.0	988.8
EDD	244.5	414.2	59.0	17.7	27.0	9.7	421.3	23.8	1,217.2

Table 6. Table of total (weighted) tardiness for jobs of each weight, provided by the different methods for the Morgan Stanley scheduling problem

3.7. Conclusions and Directions of Future Research

In this chapter, I studied the NP-hard problem of scheduling jobs with given relative priorities and deadlines on identical machines, minimizing the TWT measure. I first showed that the problem can be converted to quadratic form and then the HNN approach is a heuristic which consistently outperforms other benchmark heuristics such as the EDD and WSPT methods. Furthermore, I improved on this method by demonstrating that random perturbations to the intelligently selected starting point significantly improves the quality of the solution of the HNN approach. Finally, I studied the implementation details of the random perturbations and the HNN method to show that the methods remain applicable in real-time settings for scheduling large number of problems, PSHNN yielding a 5% improvement over the next best method, PLWPF and 7% improvement over regular HNN.

As for directions for further research, more formal methods for the selection of alpha, beta and gamma parameters could be investigated in order to further improve performance and bounds on the performance of these methods could be established, in relation to the theoretical optimum.

CHAPTER 4

SUMMARY OF RESULTS AND THESES

In this dissertation I have examined two important areas of computational finance: optimal portfolio selection and optimal resource scheduling. Each of these problems is of crucial importance in present day financial computing research and applications. Fast heuristics in optimal portfolio selection have the potential to apply these methods in real-time high-frequency algorithmic trading while optimal scheduling has the potential to speed up financial computations to save significant costs and improve transparency by having the results available sooner to be reported to financial regulators. I have selected an NP hard open problem within each area and explored polynomial time heuristic methods to provide fast solutions which outperform other benchmark methods. In each case, I have managed to come up with novel approaches which

- produce results which are superior to other heuristics found in the literature
- have low computational complexity, can be evaluated in polynomial order of time
- have practical runtime characteristics which make them applicable in real world settings

Furthermore, I managed to make a number of other contributions to the solution of each problem. In the case of the sparse, mean reverting portfolio selection I have made significant improvements to the parameter estimation of the VAR(1) and Ornstein-Uhlenbeck processes. In the TWT scheduling problem I have proven that it can be converted to a quadratic optimization problem via a nontrivial matrix algebraic mapping. I have also improved upon the standard HNN method by introducing random perturbations to an intelligently selected initial point (PSHNN method).

The achievements of numerical tests on real-world problems are presented in the following table:

Field	***Real world problem***	***Average. performance of traditional approaches***	***Average performance of the proposed new method***	***Impact on computational finance (improvement in percentage)***
Portfolio optimization	Convergence trading on US S&P 500 stock data	11.6% (S&P 500 index return)	34%	22.4%
Schedule optimization	Morgan Stanley overnight scheduling problem	24709 (LWPF performance)	22257 (PSHNN performance)	10%

Table 7. Summary achievements of numerical tests of my research work on real-world problems

Considering the above results, I have achieved the aims of the dissertation.

The overall practical impact of my work is that these problems with high complexity can now be solved approximately, in the fraction of the time required to compute the exact solution. As such, a financial services company employing these methods can provide faster and more reliable services to its clients. The algorithms presented for solving the scheduling problem can also be adopted in settings broader than computational finance, they are equally applicable to problems in computational biology, chemistry or other grid computing settings. Furthermore, some portfolio selection and trading algorithms which could only be used off-line in a low-frequency trading setting can now be used in a high-frequency, real-time environment, producing faster return on capital and therefore lowering the overall funding cost of trading. Finally, the computational improvements achieved by better scheduling do not only reduce the costs to the institutions, but also make the results of the calculations available sooner to the regulators, thereby increasing the transparency of the financial markets and serve the society as a whole.

4.1. Unified approach to complex financial modeling problems

In the course of developing these heuristic methods, I have applied a consistent approach and high level methodology which is illustrated in the below figure. This approach can be applied more broadly to challenging problems in computational finance and beyond. The basic idea is that after defining and formalizing the problem, we need to establish the theoretical complexity of the task. If it is found that the problem is non-polynomial then at first, basic heuristics must be established which yield approximate solutions to the problem. Following this, more complex and more accurate heuristics must be developed which gradually take more and more of the constraints into account and thus yield more and more accurate approximations. These must then be tested on a large number of generated and real-world problems in order to prove the viability and scalability of the approaches. These can be compared to simpler heuristics or if the problem size allows, even to the solution of exhaustive search which provides the theoretically best solution.

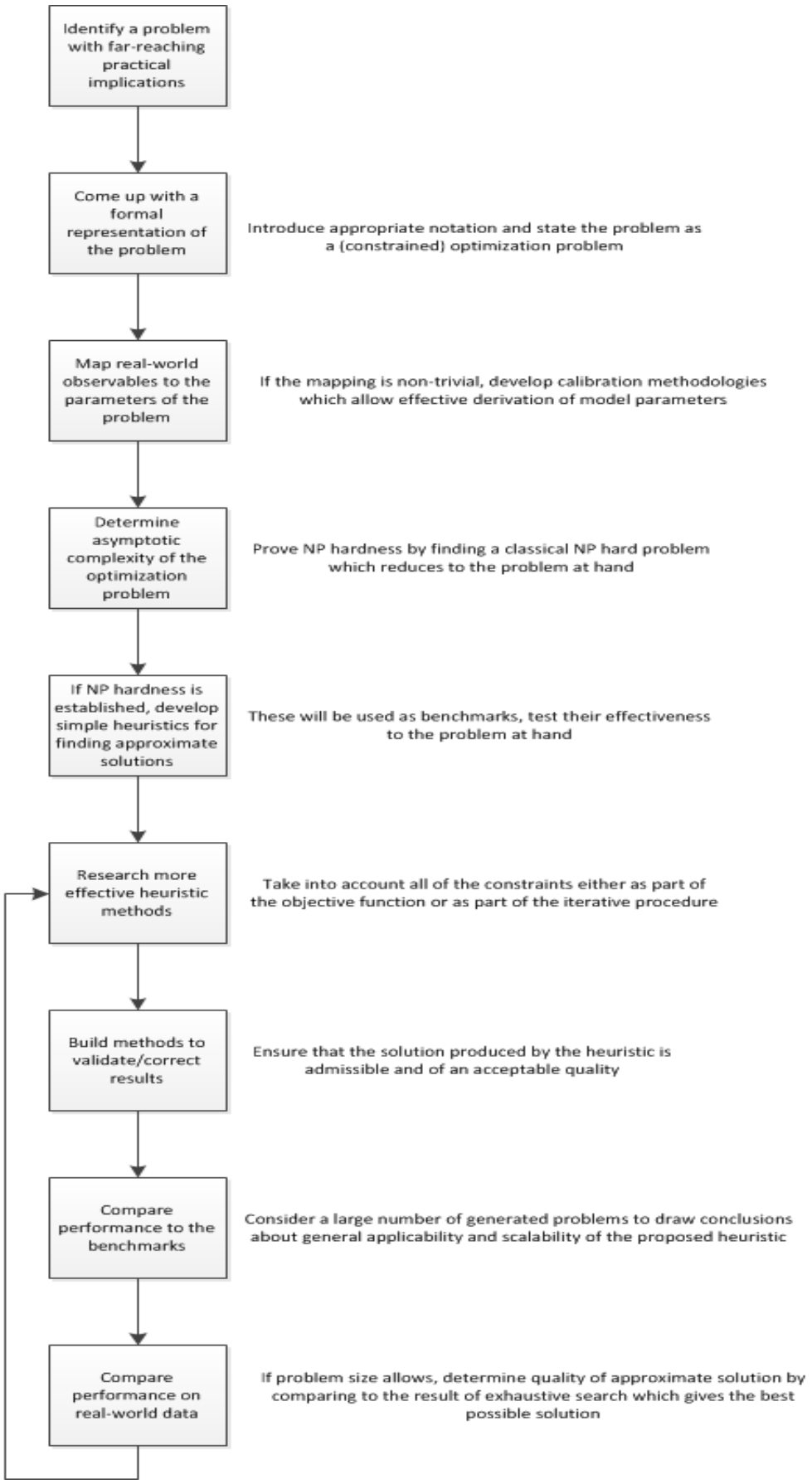

Fig. 30: Flowchart showing step-by-step unified approach to finding fast heuristic approximate solutions to complex problems in computational finance and beyond

CHAPTER 5

LIST OF REFERENCE PUBLICATIONS

5.1. Publications of the author

Journal Publications [19 points]

[1] Fogarasi, N., Levendovszky, J. (2012) A simplified approach to parameter estimation and selection of sparse, mean reverting portfolios. *Periodica Polytechnica,* 56/1, 21-28. [4 points]

[2] Fogarasi, N., Levendovszky, J. (2012) Improved parameter estimation and simple trading algorithm for sparse, mean-reverting portfolios. *Annales Univ. Sci. Budapest., Sect. Comp.,* 37, 121-144. [4 points]

[3] Fogarasi, N., Tornai, K., & Levendovszky, J. (2012) A novel Hopfield neural network approach for minimizing total weighted tardiness of jobs scheduled on identical machines. *Acta Univ. Sapientiae, Informatica*, 4/1, 48-66. [6/2=3 points]

[4] Tornai, K., Fogarasi, N., & Levendovszky, J. (2013) Improvements to the Hopfield neural network solution to the total weighted tardiness scheduling problem. *Periodica Polytechnica,* 57/2, 57-64. [4/2=2 points]

[5] Fogarasi, N., Levendovszky, J. (2012) Sparse, mean reverting portfolio selection using simulated annealing. *Algorithmic Finance,* 2/3-4, 197-211. [6 points]

Conference Presentations

[6] Fogarasi, N., Levendovszky, J. (2012) Combinatorial methods for solving the generalized eigenvalue problem with cardinality constraint for mean reverting trading. *9th Joint Conf. on Math and Comp. Sci. February 2012 Siófok, Hungary*

Technical Reports

[7] Fogarasi, N. (2003) Option pricing primer. *Technical Report*
[8] Fogarasi, N. (2003) Non-gaussian option pricing. *Technical Report*
[9] Fogarasi, N. (2003) Option pricing using neural networks. *Technical Report*

5.2. Publications connected to the dissertation

[10] Adcock, C.J., and Meade, N. (1994) A simple algorithm to incorporate transaction costs in quadratic optimization. *European Journal of Operational Research,* 79:85-94.

[11] Akyol, D.E., & Bayhan, G.M. (2008) Multi-machine earliness and tardiness scheduling problem: an interconnected neural network approach. *International Journal of Advanced Manufacturing Technology*, Springer-Verlag London Limited 37:576-588

[12] Anagnostopoulos, K.P., and Mamanis, G. (2011) The mean-variance cardinality constrained portfolio optimization problem: An experimental evaluation of five multiobjective evolutionary algorithms.. *Expert Systems with Applications,* 38:14208-14217.

[13] Armananzas, R., and Lozano, J. A. (2005). A multiobjective approach to the portfolio optimization problem. In *IEEE congress on evolutionary computation* 1:1388–1395.

[14] Armentano, V.A., & Yamashita, D.S. (2000) Taboo search for scheduling on identical parallel machines to minimize mean tardiness. *Journal of Intelligent Manufacturing*, 11, 453-460.

[15] Arnone, S., Loraschi, A., and Tettamanzi, A. (1993) A genetic approach to portfolio selection. *Neural Network World,* 6:597-604.

[16] Azizoglu, M., & Kirca, O. (1998) Tardiness minimization on parallel machines. *International Journal of Production Economics,* 55, 163-168.

[17] Banerjee, O., El Ghaoui, L., d'Aspermont A. (2008) Model Selection Through Sparse Maximum Likelihood Estimation. *Journal of Machine Learning Research,* 9: 485-516

[18] Barraud, A.Y., (1977) A numerical algorithm to solve A XA - X = Q, *IEEE Trans. Auto. Contr.*, AC-22, 883-885

[19] Biskup, D., Herrman, J., & Gupta, J.N.D. (2008) Scheduling identical parallel machines to minimize total tardiness. *International Journal of Production Economics,* 115, 134-142

[20] Boguslavsky, M., Boguslavskaya, E. (2004) Arbitrage Under Power. *Risk Magazine* 2004 June, 69-73

[21] Box, G.E., Tiao, G.C. (1977) A canonical analysis of multiple time series. *Biometrika* 64(2), 355-365

[22] Brucker, P. (2007) *Scheduling Algorithms*. Fifth Edition, New York: Springer

[23] Chang, T.-J., Meade, N., Beasley, J.E. and Sharaiha, Y.M. (2000). Heuristics for cardinality constrained portfolio optimisation. *Computers and Operational Research*, 27: 1271–1302.

[24] Chen, L.and Aihara, K. (1995) Chaotic Simulated Annealing by a Neural Network Model with Transient Chaos *Neural Networks* 8(6): 915-930

[25] d'Aspremont, A, Banerjee, O., El Ghaoui, L.(2008) First-Order Methods for Sparce Covariance Selection. *SIAM Journal on Matrix Analysis and its Applications,* 30(1) 56-66

[26] d'Aspremont, A.(2011) Identifying small mean-reverting portfolios. *Quantitative Finance,* 11:3, 351-364

[27] Dempster, A (1972) Covariance selection. *Biometrics* 28, 157-175

[28] Di Tollo, G., and Rolli, A. (2008) Metaheuristics for the portfolio selection problem. *International Journal of Operations Research*, 5:13–35

[29] Dixit, A.K., Pindyck, R.S. (1994) Investment Under Uncertainty, *Princeton University Press,* Princeton, NJ.

[30] Du, J. & Leung, J. Y.-T. (1990) Minimizing total tardiness on one machine is NP-hard. *Math. Oper. Res*., 15(3), 483-495.

[31] Dogramaci, A., & Surkis, J. (1979) Evaluation of a heuristic for scheduling independent jobs on parallel identical processors. *Management Science,* 25(12), 1208-1216.

[32] Elton, E.J., Gruber, M.J., Brown, S.J., and Goetzmann W.N. *Modern portfolio theory and investment analysis(7th ed.)* Wiley (2007)

[33] Fama, E., French, K.(1988) Permanent and Temporary Components of Stock Prices. *The Journal of Political Economy* 96(2), 246-273

[34] Farber R. (2011). *CUDA Application design and development*. Morgan Kaufmann 1st ed.

[35] Feiring, BR., Wong, WL, Poon, M., and Chan Y.C. (1994) Portfolio selection in downside risk optimization approach – application to the Hong Kong stock market. *International Journal of Systems Science* 25:1921-1929.

[36] Fernández, A., and Gómez, S. (2007) Portfolio selection using neural networks. *Computers & Operations Research,* 34:1177–1191

[37] Geman, S. and Geman, D. (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence,* 6:721-741

[38] Gillespie, D.T. (1996) Exact numerical simulationof the Ornstein-Uhlenbeck process and its integral. *Physical review E* 54:2, 2084-2091

[39] Grant, Jeremy (2010) High-frequency trading: Up against a bandsaw. *Financial Times* 2010 Sep 2.

[40] Guinet, A. (1995) Scheduling independent jobs on uniform parallel machines to minimize tardiness criteria. *Journal of Intelligent Manufacturing*, 6, 95-103.

[41] Haykin, S. (2008) *Neural Networks and Learning Machines*. 3rd Edition, Prentice Hall

[42] Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities in *Proceedings of the National Academy of Sciences of the USA*

[43] Ingber, L. (1993) Simulated annealing: practice versus theory. *Math. Comput. Modelling,* 18 29-57

[44] Ito, K. Stochastic Integral. (1944) *Proc. Imperial Acad. Tokyo* 20, 519-524

[45] Johnson, D.S., Aragon, C. R., McGeoch, L.A. and Schevon, C. (1989) Optimization by simulated annealing: an experimental evaluation. Part I, graph partitioning. *Operat. Res.* 37, 865-892

[46] Johnson, D.S., Aragon, C.R., McGeoch, L. A. and Schevon, C. (1991) Optimization by simulated annealing: an experimental evaluation; Part II, graph coloring and number partitioning. *Operat. Res.* 39, 378-406

[47] Kirk, D. B. & Hwu W. W. (2010). Programming massively parallel processors: A hands-on approach. Morgan Kaufman.

[48] Kirkpatrick, S., Gelatt, C.D.and Vecchi, M.P. (1983) Optimization by Simulated Annealing, *Science*, 220(4598), 671-680

[49] Konno, H. and Yamazaki, H. (1991) Mean/absolute devation portfolio optimiziation model and its applications to Tokyo Stock Market. *Management Science* 37: 519-531

[50] Koulamas, C.P. (1994) The total tardiness problem: review and extensions. *Operations Research,* 42, 1025-1041.

[51] László, E., Tornai, K., Treplan, G., & Levendovszky, J. (2011) Novel load balancing scheduling algorithms for Wireless Sensor Networks. *The Fourth International Conference on Communication Theory*, *Reliability, and Quality of Service*, April 17-22, 2011

[52] Lawler, E. L. (1977) A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*., 1, 331-342

[53] Lawler, E. L. (1981) Preemptive scheduling of precedence-constrained jobs on parallel machines, deterministic and stochastic scheduling. *Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems*

[54] Levendovszky, J., Mommaerts, W., & van der Meulen, E.C (1992) Hysteretic Neural Networks for global optimization of quadratic forms. *Neural Network World,* 2(5):475-496

[55] Levendovszky, J., Olah, A., Tornai, K., & Treplan, G. (2011) Novel load balancing algorithms ensuring uniform packet loss probabilities for WSN. 2011 IEEE 73rd Vehicular Technology Conference Budapest

[56] Liang, Li (2011). Parallel implementations of Hopfield neural networks on GPU. Tech Rep. Dépót Universitaire de Mémoires Apres Soutenance.

[57] Loraschi, A., Tettamanzi, A., Tomassini, M., and Verda, P. (1995) Distributed genetic algorithms with an application to portfolio selection problems. In: Pearson, D.W., Steele, N.C. and Albrecht, R.F. (Editors). *Artificial neural nets and genetic algorithms,* 384-387.

[58] Maheswaran, R., Ponnambalam, S.G., Nithin S. D., Ramkumar, A.S. (2004) Hopfield Neural Network Approach for Single Machine Scheduling Problem. *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems*

[59] Mandziuk, J. (1996) Solving the Travelling Salesman Problem with a Hopfield - type neural network. *Demonstratio Mathematica*, Vol. 29(1)

[60] Mandziuk, J.,& Macukow, B. (1992) A neural network designed to solve the N-Queens Problem. *Biological Cybernetics,* Vol 66(4)

[61] Manzan, S. (2007) Nonlinear Mean Reversion in Stock Prices. *Quantitative and Qualitative Analysis in Social Sciences*, 1(3), 1-20

[62] Markowitz, H. (1952) Portfolio Selection. *The Journal of Finance* 7:1,77-91

[63] Martel, C. (1982) Preemptive Scheduling with Releast Times. *Deadlines and Due Dates Journal of the Association for Computing Machinery*. Vol 29, No 3, 812-829

[64] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H and Teller, E. (1953) Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics,* 21(6), 1087-1092

[65] Murty, K. G. (1988) Linear complementarity, linear and nonlinear programming. Heldermann Springer-Verlag

[66] B. Naderi, M. Zandieh & M.A.H.A. Shirazi, (2009). Modeling and scheduling a case of flexible flowshops: Total weighted tardiness minimization. *Computers & Industrial Engineering*, **57** (4): 1258-1267.

[67] Natarjan, B.K. (1995) Sparse approximate solutions to linear systems. *SIAM J. Comput.* 24(2), 227-234

[68] Nocedal, J., & Wright, S. J. (2006) *Numerical Optimization.* Springer-Verlag

[69] Nourani, Y. and Andersen, B. (1998) A comparison of simulated anneling cooling strategies. *Journal of Physics A: Mathematical and General* 31. 8373-8385

[70] Ornstein, L. S. and Uhlenbeck, G.E. (1930) On the Theory of the Brownian Motion. *Physical Review* 36(5), 823

[71] Penzl, T., (1998) Numerical solution of generalized Lyapunov equations. *Advances in Comp. Math.*, 8, 33-48

[72] Pinedo, M. (2008) *Scheduling – Theory, Algorithms and Systems*. Third Edition, New York: Springer

[73] Poterba, J.M., Summers, L.H. (1988) Mean reversion in stock prices: Evidence and implications. *Journal of Financial Economics* 22(1), 27-59

[74] Rachamadugu, R.M.V., & Morton, T.E. (1983) Myopic heuristics for the weighted tardiness problem on identical parallel machines.*Tech Report CMU-RI-TR-83-17,* Carnegie-Melon University, the Robotics Institute.

[75] Rothman, A., Bickel, P., Levina, E., Zhu, J. (2008). Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics,* 2: 494-515

[76] Sahni, S. (1974) Computationally related problems. *SIAM Journal on Computing*, 3.

[77] Sahni S. (1979) Preemprive Scheduling with Due Dates. *Operations Research*, 27(5), 925-934

[78] Salamon, P., Sibani, P. and Frost, R. (2002) Facts, Conjectures, and Improvements for Simulated Annealing. SIAM Monographs on Mathematical Modeling and Computation

[79] Sen, T., Sulek, J.M., Dileepan, P. (2003) Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. *International Journal of Production Economics*, 83, 1-12.

[80] Seydel, R. U. (2006) *Tools for computational finance*. Springer, 3rd edition

[81] Sims, C. (1980) Macroeconomics and reality. *Econometrika,* 48 (1):1-48.

[82] Streichert, F., Ulmer, H., & Zell, A. (2003). Evolutionary algorithms and the cardinality constrained portfolio optimization problem. In: *Selected papers of the international conference on operations research* (pp. 253–260)

[83] Smith, W. (2010). On the simulation and estimation of mean-reverting Ornstein-Uhlenbeck process. *Technical Report, February 2010.*

[84] Treplán, G., Tornai, K., & Levendovszky, J. (2011) Quadratic Programming for TDMA Scheduling in Wireless Sensor Networks. *Accepted by International Journal of Distributed Sensor Networks*

[85] Yoshomito, A. (1996) The mean-variance approach to portfolio optimization subject to transaction costs. *Journal of the Operations Research Society of Japan,* 39:99-117.

[86] Zhi-quan L.,Wing-kin M., So, A.M.-C, Yinyu Y., & Shuzhong Z. (2010) Semidefinite Relaxation of Quadratic Optimization Problems. *Signal Processing Magazine*, IEEE, 4(27)