

HEURISTIC APPROACHES TO SOLVING MULTIOBJECTIVE SCHEDULING OPTIMISATION PROBLEMS

Ezeani M I¹; Agbasi K C¹; Osita E G¹; Okonkwo R O¹; Nnanwa C P²

¹Department of Computer Science, Nnamdi Azikiwe University, Awka

²Department of Mathematics, Nnamdi Azikiwe University, Awka

Abstract

In recent years, different heuristic and hyperheuristic optimization approaches involving natural computing techniques have been suggested to solve multiobjective combinatorial optimization problems. These techniques have been applied – individually and in hybrid forms – with relative success to scheduling problems. This work attempts to review the performance of the three common techniques often applied in scheduling problems: Genetic Algorithms (GA), Simulated Annealing (SA) and Tabu Search (TS).

Keyword: scheduling, optimisation, genetic algorithms, simulated annealing, tabu search.

1.0 Introduction

Over the last five decades, scheduling problems feature more prominently in Operations Research (OR) areas than in computing because manual or mathematically based solutions were sufficient [1, 2, 3]. However, the growing complexity of the scheduling problems and the advances in the development of high speed computing devices fuelled the desire to push the boundaries of knowledge in this area [4, 5]. In this work, a review of the three heuristic¹ techniques – genetic algorithms (GA), simulated annealing (SA) and Tabu Search (TS) – that are often applied in solving scheduling problems is presented. This introduction gives an overview of the concept and main principles behind scheduling.

1.1 Overview of scheduling

Scheduling is a very important concept, not just in computing and business management, but to life in general. Sims [6] rightly observes that:

"People face scheduling problems and opportunities everyday: at the airport,

someone is responsible for assigning planes to gates; at a plumbing service, someone is in charge of assigning repairmen to service calls; in a manufacturing facility, someone is in charge of assigning jobs to machines, in a call centre, someone is in charge of assigning certain calls to specifically skilled personnel".

The above is obviously not exhaustive but gives a clear idea of how involved people are with scheduling in their day to day activities. Scheduling involves developing and managing a *plan*² to achieve an objective in solving the general problem of satisfying time and resource constraints between a number of tasks [7, 6, 8]. Rivreau [7] captures, in his simple definition, what the business of scheduling actually involves:

To schedule is to fix jobs (activities) executions:

- *by assigning resources to these jobs*
- *by fixing their starting times*
- *and by verifying the constraints*
- *in order to optimise one or several fixed criteria*

However, the simplicity of the definition did not undermine the complexity of the problem. Burke *et al.* [9] and Filho *et*

¹ Heuristic is a "speculative formulation serving as a guide in the ... solution of a problem" [A]. A hyperheuristic manages other heuristics for different problems while a metaheuristic manages one other heuristic [9, 34]

² Plan is defined as a scheme, program, or method worked out beforehand for the accomplishment of an objective [16, 33]

al. [10] admit that it could be challenging to *avoid conflicts* (of objectives) and *satisfy side-constraints* over a *limited time period*. Some of the approaches used to solving are looked at in the following sections.

1.2 Approaches to solving scheduling problems

Scheduling problems are typically *multiobjective combinatorial optimisation* problems [11,12,13,14,3,15] (to be discussed later) and vigorous research activities have been going on in this area for decades [7]. Some of the approaches to solving them include: manual, mathematical models, advanced algorithms. AI methods and heuristics [7, 8]. Abraham *et al* [4] however, observed that scheduling problems are so rich that “*no single scheduling method is sufficient*”. So several techniques (including heuristic approaches), that worked well on classical scheduling problem models, have been suggested by researchers [9,16,12,5,3]. Heuristic approaches are also generally known to have produced “high-quality” results [16].

1.3 Application areas

The application of scheduling spans a wide variety of areas ranging from workforce scheduling and timetabling [15,35,44] to robot control systems, intelligent agents, grid computing and earth observing satellites [1, 30,31,32]. These areas exist in various domains like the education, military, health, transportation, manufacturing, government, business and economy [20,34]. A few particular examples from the works reviewed will be presented later in this work.

The rest of this paper is organised as follows. *Section 2* deals with the nature of scheduling problems, why they are difficult and presents an overview of multiobjective optimisation concepts; *section 3* looks at the underlying ideas behind TS, GA and SA; *section 4*

overviews some of the published scheduling “systems” and models developed with any or a combinations of the techniques and finally, in *section 5* concludes with a discussion on the future direction of research in this area.

2.0 Problem Definition

Wolfe [8] asserts that most scheduling problems deal with issues involving assigning activities or tasks to resources, satisfying the constraints and optimizing the objective function. In their complex forms, they typify *NP-complete*³ problems [20, 9, 19, 3, 21] and they also occur in different forms but generally have the same outlook and structure.

2.1 The main components of a scheduling problem model

In their individual analysis, Leeuwen [22] and Sims [6] identified a number of components that make up a typical scheduling problem model:

1. *the properties of the each jobs* (or tasks/activities) e.g. the processing time, deadline, release time, the priority
2. *the properties of the machines (resources)* e.g. identical or different (in capacity may be), availability, parallel, flow shop or job shop⁴
3. *the properties of the processing* e.g. constraints, penalties, on-line, off-line
4. *the objective of the scheduling* i.e. the ‘cost’ of the schedule is to be minimised

The task of defining the rules that are used to assign the task to the resources is obviously the most important aspect in the design of scheduling systems [6]. These rules are at the lowest level very simple e.g. *get the Earliest Due Date job* or *get the Shortest Processing Time job*. But real life domains which have rules that are

³ “NP-complete refers to a subclass of the hardest problems in NP” see [A] for details

⁴ Flow shop models have machines in series while Job shop models are processed according to a specified route through the set of machine [22].

complex combinations of the simple ones are simply difficult to model [6, 8].

2.2 Why scheduling problems are difficult

In real life, complex rules and decision making processes are usually employed to generate feasible schedules. Wolfe [8] attributes the difficulty in modelling real life scheduling problems to the following:

- complexity* (huge number of options, combinatorial explosion),
- objectives or criteria* (vague, ambiguous, multiple and confliction),
- uncertainties* (new orders/priorities, unexpected events),
- domain specificity* (hard to apply existing techniques).

As with every multiobjective combinatorial optimisation problem, the search space is usually large and complex and the domain difficult to model [12, 8].

2.3 Multi-objective optimisation

The development of acceptable and workable schedules requires, to a large extent, the use of optimization techniques [4, 23, 12, 13, 24]. According to Vincent [25], optimisation involves the process of improvement of the quality of a solution to a problem. It usually begins with an *initial* candidate solution and then iteratively searches through the problem's solution space for a better solution which, if and when found, replaces the existing solution. So this approach requires a function – the *objective function* – that evaluates the quality of the solutions (new and old) in order to make a reasonable decision and is formally defined as given below [25]:

$$f: X \subset \mathbb{R}^n \rightarrow \mathbb{R}$$

where X is a user-defined bounded set of valid (or desirable) inputs and f is the objective function (refer to [13,24,25] for further definitions)

Above describes a typical single objective problem (because it shows only one objective function). Where a single perfect (or Utopian) solution exists – the global optimum – it can be found depending on

the search strategy [24]. However, in reality we most often need to simultaneously optimise multiple, possibly conflicting objective functions. A suitable solution often compromises optimal performance of any one of the objectives to achieve *acceptable* (or *Pareto optimal*⁵) solutions in all objective dimensions [24]. In practice though, these objectives need to be recombined to form a single objective function that represents the component functions with a redefinition of the required performance level in order to be able to select good solution from a family of alternatives which are otherwise considered equivalent [13, 24]. Generally, similar approaches to search for a better solution in the solution space apply.

A formal definition of a typical multi-objective problem [24] is given as follow:

$$f(x) = (f_1(x), \dots, f_n(x))$$

$$f(x) = (f_k(x)); k = 1, \dots, n$$

Where x = a vector solution variable

$f_k(x)$ = a component objective function

$f(x)$ = the combination of the component functions.

And then a solution vector, $x_u \in U$, is said to be Pareto optimal if and only if there is no $x_v \in U$ for which $f(x_v)$ dominates $f(x_u)$ [24]. There could be a set of alternative solutions and so choosing any one of them is often subjective.

So in different kinds of scheduling problems, these issues are of concern in selecting the techniques that will efficiently produce admissible set of solutions.

2.4 Heuristic approaches to solving scheduling problems

Any scheduling algorithm (or *scheduling policy* [22]) is expected to satisfy the *hard*⁶ constraints by producing a feasible

⁵ The set of Pareto optimal decision vectors are such that any decrease in one objective must cause a simultaneous increase in at least one other objective [15, 22, 38].

⁶ Hard constraints are rigid and unbending and define the search space; soft constraints are flexible and define the quality of the solution [16, 8].

solution [13, 19, 8]. However, the *quality* of the algorithm is a function of how *fast* it finds the optimal solution, the *reliability*, *simplicity* and *portability* of the algorithm, and of course how well it satisfies the soft constraints [8].

Heuristic techniques have therefore been successfully applied because they borrow the human tendency toward a useful solution and as such solving the problem by covering only a subset of the search space [25]. Efficiency is most often guaranteed and in many cases, the concepts behind them are amazingly simple. Genetic algorithms (modelled on human evolution), Tabu Search (a *neighbourhood* technique) and Simulated Annealing (based on thermodynamics) have performed very well as will be discussed in later sections [4, 26].

In such complex domains as combinatorial optimisation problems, the need for elegant, robust and adaptable techniques cannot be over emphasised. These techniques work very well and are yet amenable to investigation and refinement [27] and so the future is likely to see huge improvements on these techniques and their hybrids especially as the hardware technologies garner more processing power [4,5].

Some of the problems that have provided testing grounds for these techniques and engaged researchers attention are the classic *Travelling Salesman Problem* (TSP), the *Job Shop Scheduling Problems* (JSP), *Timetabling* and *Transportation* problems.

However, researchers' attentions are recently being drawn to the "*surprising degree of effectiveness and generality demonstrated by heuristic approaches*" to these classes of problems [4]. Across the globe, different scheduling systems are modelled purely on simple search algorithms which harness the richness of natural heuristic search methods [4,28,11,29,27,30]. Their application to some mission critical projects with success further reinforces their reliability

[1,27,30,31]. Currently, research resources are being invested in this area with the hope of improving the quality of the techniques in terms of adaptability and generality [23, 28, 9, 3].

3.0 *Review of selected techniques*

Heuristics techniques are a bundle of possibilities. The following section presents an overview of three of the most common heuristic techniques that are usually applied in solving scheduling problems.

3.1 **Tabu Search (TS)**

Tabu Search is one of the most common heuristic approaches to solving scheduling problems and combinatorial optimization problems in general [32, 33]. It had been applied to several NP-hard problems and has proven capacity to give good results [15, 21].

The origin of TS, dates back to the late 70s but was presented first in 1986 by Fred Glover [34,32,33,19]. It is a neighbourhood search algorithm which begins by generating an initial, i , to the problem. It then looks around the neighbourhood of i , $N(i)$, for a subset of the solution space and picks the best contestant, j , to replace i (even if j is worse than i). This process is repeated with the neighbourhood adjusted for the current solution [4,33,19] until a certain stopping criterion is met. Because the neighbourhood changes with the current solution, TS has often been referred to as a "*dynamic neighbourhood search technique*" [33].

In TS, there is a risk of cycling through a set of solution and getting trapped in a local optima [33] and this could be avoided by forbidding recently visited solutions from being selected. These solutions are kept in a list called the Tabu List, T which operates based on FIFO (First In First Out) principle – i.e. solutions leave the list in the order they arrived. Also, a variation of this list exists, which holds not the solutions but the *moves* made from i (the current solution)

to j (the next solution) such that the neighbourhood is defined by [33]:

$$N(i) = \{j \in M \mid M(i) \text{ such that } j = i \oplus m\}.$$

So j , a neighbour of i , can be obtained if a move, m , operates on i . Some TS variations often consider approaches the reverse moves, m^{-1} , such that $i = i \oplus m \oplus m^{-1}$ in guiding the search to more promising areas [4,33,19].

TS uses the *aspiration level criterion* (or *aspiration level condition*) to determine when these restriction rules could be relaxed [33, 19, 21]. For instance, if a move is likely to get a better solution than the overall best, it should be considered even if the move is tabu⁷.

Well structured TS algorithms strike a good balance between *intensification* (concentrating more in a region in the search space that promises good solutions) and *diversification* (spreading across the entire search space). Intensification may bias the algorithm toward solutions with common features with the current solution while diversification tends toward solution not so related to current solution. A good balance between the two, depending on the nature of the problem, is often an issue that may affect how well the algorithm will perform. So the modified objective function, F , [33] thus becomes:

$$F = f + \text{Intensification} + \text{Diversification}$$

A typical TS algorithm reads as follows [33]

1. Choose an initial solution i in S . Set $i^* = i$ and $k = 0$.
2. Set $k = k + 1$ and generate a subset V^* of solution in $N(i, k)$ based on the move operators and the Aspiration function.
3. Choose a best j in V^* (with respect to f or to the function F) and set $i = j$.
4. If $f(i) < f(i^*)$ then set $i^* = i$.
5. Update tabu and aspiration conditions.
6. If a stopping condition is met then stop. Else go to Step 2.

⁷ Recent moves are generally forbidden i.e. they are declared tabu.

3.2 Genetic Algorithms (GAs)

Genetic Algorithms (GAs) are well known function optimisers which belong to the group of computational models inspired by the Darwinian Theory of Evolution [35, 10, 25, 36, 37, 21]. The concept was originally proposed by John Holland in 1975[37]. GAs work with a population of individual data structures often encoded in a chromosome-like form. These individuals in the population are a sample of the solution space currently known to the GA [42].

A GA typically begins by randomly generating the individuals that make up the *initial population*. It then iteratively goes through the *selection* process to pick pairs of individuals for the *intermediate population* (often referred to as the *mating pool*). The next stage involves the probabilistic operations of *crossover* and *mutation* which, in each case, produce an offspring (new individual) to make up a new generation of individuals. The last stage is the replacement of the current population with the new. The processes – selection, crossover + mutation, replacement – are repeated until a stopping condition is reached e.g. at the end of a given number of generations or when an acceptable solution quality is reached. The overall (or so far) best solution is retained and the average quality of the entire population of the entire population of solutions improves through the evolving generations of individuals.

Experimenting with the GA models, researchers have over the years suggested different variations of the original canonical model proposed by Holland. Whitley [37] argues that the term GA in a broader sense actually refers to “any population-based model that uses the selection and recombination [crossover] to generate new sample points in the search space”.

Good selection techniques offer higher chances of selection to individuals with

above-average fitness⁸; crossover attempts to preserve the good traits in the new offspring while mutation basically preserves diversity in the system [25, 37]. The fitness of the individual is the probability of selecting it from the population for mating based on its value from the *evaluation* (or the objective) function [37].

They all seek a balanced level of *exploration* and *exploitation* in the search space. The basic structure of a GA is presented as follows [26]:

1. Generate $P(0)$
2. $t := 0$
3. while not Termination_Criterion($P(t)$)
4. do
5. Evaluate($P(t)$)
6. $P'(t) := \text{Selection}(P(t))$
7. $P'(t) := \text{Recombination}(P'(t))$
8. $P'(t) := \text{Mutation}(P'(t))$
9. $P(t+1) := \text{Replace}(P(t), P'(t))$
10. $t := t+1$
11. return Best_Solution_Found

3.3 Simulated Annealing (SA)

Simulated Annealing borrows its operational idea from thermodynamics. Annealing refers to the process by which high energy particles that form a system gradually lose energy due to cooling and get to very low energy states [17,25]. It is a generalisation of the Metropolis Monte Carlo integration algorithm by Kirkpatrick [17,25,21].

The four basic components of SA algorithms are: the Acceptance Function, the distribution of the next candidate point, the cooling schedule and the stopping rule. The basic algorithm, according to Locatelli [17] works as follows:

0. randomly generate the initial solution: $\in X$
1. $z_0 = \{x_0\}$ // Keep all visited solutions in the set z_0
2. Generate a new candidate, y_{k+1} from the 'next candidate distribution D '

3. randomly generate a value, p , in the $[0,1]$

$$x_{k+1} = \begin{cases} y_{k+1} & \text{if } p \leq A(x_k, y_{k+1}, t_k) \\ x_k & \text{otherwise} \end{cases}$$

$A = \text{Acceptance function with values between 0 and 1.}$

$t_k = \text{temperature parameter}$

4. Add y_{k+1} to the set z ($z_{k+1} = z_k \cup \{y_{k+1}\}$)
5. New temperature, t_{k+1} , from the current set Z [using a defined cooling schedule]
6. Repeat (1 – 6) until a 'stopping criterion' is met.

Locatelli [17] also explains that the Acceptance Function informs the decision to accept new candidate points or otherwise. A typical acceptance function (also known as the Metropolis function) is a given below:

$$A(x, y, t) = \min [1, (\exp - f(y) - f(x))/t]$$

Tuning an SA to good performance largely depends on the choice of the cooling schedule and of the next candidate distribution [17].

Again the stopping rule that guarantees stopping at the global optimum, at times, appears to be a difficult with SA. Some heuristic approaches can be adopted like when a given number of iterations are reached or when there is no acceptance for a number of times or when an acceptable objective function value is got.

4.0 Applications of GA, TS, SA techniques to scheduling problems

Researchers have extensively applied GA, TS, SA and their hybrids in solving classic scheduling problems. This paper reviews some of the works done in this field.

Deb K *et al.* [14] implemented a fast elitist non-dominated GAs (NSGA-II) for solving multiobjective optimisation problems. Their earlier on this, the NSGA, had been criticized for high computational complexity, lack of elitism and the need for parameter specification. The newly

⁸ The fitness function is often defined in terms of the individuals chance of being selected and not just in its quality[36]

proposed parameter-less approach with elitism did outperform other approaches to similar problems.

Coates *et al.* [29] applied GA to develop their Real-Time Coordinated Scheduling for an agent-oriented system – Design Coordination System (DCS). They claimed their work reduced re-scheduling time by half.

Filho and Lorena [10] successfully implemented a GA based “*conflict-free*” school time table with a specialised mutation operator. Similarly, Alkan and Özcan [28], in their work, described a memetic algorithm⁹ that dealt with hard constraints and also introduced operators such as “*violation directed mutation*” and “*hierarchical hill climbing*” which have promising features for search algorithms. Also, Weare *et al.* [38] worked on “*highly constrained*” time table problems and developed a “*powerful*” hybrid GA. Burke and Petrovic [16] further demonstrated the efficiency of heuristic approaches in their in this area.

Alba *et al.* [26] observed that a work planning problem originally formulated as a mixed integer problem and solved with Cplex could not deal with “*unstructured*” problem. They described and implemented a Parallel GA (PGA) which performed faster but was not as accurate as the Cplex. Chen and Liu [11] proposed a fast and efficient Multi-level Genetic Assembly Sequence Planner which proved reliable on all the tested problems.

Han, Kendall and Cowling [39] presented a hyperGA with an adaptive length chromosome to solve Trainer Scheduling problem. They reported success in applying their work to geographically distributed training staff.

Schaerf [3] presented a rich survey of automated time tabling in 1999 highlighting the explosion of possibilities with heuristic approaches. He observed

that exact methods can only be applied in small cases while in real instances, “*only heuristic methods are feasible*”. In a related work, Ernst *et al.* [] reviewed about 700 literatures of personnel scheduling and rostering. In his work, different heuristic approaches account for about 70% of all the methods used while the GA:SA:TS ratio was at 7:5:4 (see p41). Also Coello Coello [12] worked on a critical survey of GA-Based multiobjective optimisation techniques and their underlying motivations, problems and possible ways to improve performance. Another survey was reported by Cheng *et al.* [12] who evaluated some of the recent hybrid genetic approaches to solving Job Shop Scheduling, highlighted some of the weaknesses of GA and techniques to overcome them. Locatelli's paper [17] was more of a useful handy tutorial on SA and an overview of the use of SA in solving continuous global optimisation problems. Wang and Zheng [36] successfully implemented a hybrid GA-SA, using SA as the mutation operator while in his work, Vincent [25] applied GA to solving the maximization variant of the classical Rastrigin function and also provided a rich overview of heuristic structured systems. Ciesielski and Scerri [1] presented a GA based “*anytime algorithm*” for scheduling aircraft landing times – a real time system. They argue that the inherent parallel structure of GA gives it considerable advantage over other methods in such problems.

Abraham *et al.* [4] demonstrated the use of these heuristics in resource scheduling on a computational grid. They reported that both hybrids GA-SA and GA-TS gave better results than GA. While in GA-SA they harnessed both the convergence properties in SA and parallelisation in GA; GA-TS, mutation operator was replaced with TS.

Aickelin [23] reported a fast generalised GA based set covering problem solver, however he admitted that his algorithm is outperformed by a problem specific heuristic.

⁹ Memetic algorithms are an extension of GAs, based on a model of how ideas evolve [28], they are considered as a kind of GA in this work.

Watson *et al.* [15] in their attempt to “demystify” TS, discussed various issues related to the key theoretical aspects of TS in their demonstration of the application of TS to the classical Job Shop Problem.

Several papers were published on the application of TS and hybrids to scheduling some of which include: De Bontridder [32] who worked on minimizing the total weighted tardiness in a JSS¹⁰ problem and suggested that TS could be effective in terms of *time* and *solution quality*. In their work, Ponnambalam *et al.* [19] implemented “*adjacent pairwise interchange*” method on TS to solve JSS problem and got results comparable to those of GA and SA on the same problem.

Also, Burke, Kendall, and Soubeiga [34] investigated the use of TS as a hyper heuristic maintaining and managing a number of other heuristics for a timetabling problem. They asserted that a reasonable solution takes into account the rational balance between solution quality, time and computational cost. Wangtong *et al.* [21] reported that TS with “*penalty reward*”¹¹ is superior to GA and SA based on their comparison of the three heuristic approaches implemented on the task partitioning and task scheduling problems. Wolfe and Sorensen [31] highlighted (as in [34]) the fact that a good algorithm depends on the choice that needs to be made. They compared GA with Priority Dispatch (PD) and Look Ahead algorithms on their Earth Observing Satellite (EOS) scheduling problem and observed that GA gave the best result but was much slower than the others while PD was simple and fast but can sometimes give bad results. Also Globus *et al.* [27] and Globus *et al.* [30] demonstrated in their separate but related models hypothesize that GAs can be effectively applied to EOS scheduling problems which is a typical complex, highly constrained domain.

As can be seen from the works above, heuristic approaches to scheduling has recorded huge success and remarkably provides a wide horizon for researchers to explore.

5.0 Conclusion and Future Directions

It is common knowledge that heuristic approaches produce “*high quality*” solutions [16]. Clearly, as observed by Hertz *et al.* [33], this approach has huge potentials especially with the refinement and hybridization of these heuristic techniques.

Researchers believe that no one of the techniques reviewed above – TS, GA and SA – is comprehensive on its own and suitable for various kinds of problems [4, 8, 40]. A well harnessed hybrid often performs better than one technique because it tends to take advantage of the strong points of the component techniques. GAs are arguably more popular than the other two, probably because of their flexibility and inherent parallel structure [35,36,37]. However, Cheng *et al.* [12] observed that GAs are not very good in “*fine tuning*” solutions around the optima. Also, GAs occasionally suffer from premature convergence, i.e. individuals becoming too similar to allow productive search to continue [28, 36]. Again, in view of cost, they are computationally inefficient due to the large memory required to manage the populations of solutions [4, 41, 21]. TS and SA are both neighbourhood search techniques and are better used as a meta-heuristics on other heuristics. They deal with only one solution at any time and are therefore computationally more efficient than a GA. However, the nature of the problem and the time available to solve it will suggest which technique is a better choice [31, 34]. The future is likely to concentrate on some of the key challenges pointed out by Lee *et al.* [16]:

- *Schemes that are easily formulated / customised to support variations*
- *Generalised methodology that cuts across problems*

¹⁰ JSS actually stands for Job Shop Scheduling

¹¹ *Penalty Reward* is a frequency penalisation strategy used in [21] to implement *diversification* and *intensification*.

- *Intelligent scheduling systems that can capture knowledge into themselves*
- *Effective constraint relaxation strategies*

However, as these techniques themselves evolve, the core differences between them dissolve and the future may see heuristics based on them, but more or less away from their individual original structures.

References

- [1] Ciesielski V, Scerri P 1997, *An "Anytime Algorithm For Scheduling Of Aircraft Landing Times Using Genetic Algorithms"*, Australian Journal of Intelligent Information Processing Systems, 4(3/4):206-213, Spring/Summer (Southern Hemisphere) 1997
- [2] Gueret, C. *et al.* 1995, "Building University Timetables Using Constraint Logic Programming", The ACM Digital Library/The Guide - Lecture Notes In Computer Science; Vol. 1153: First Int'l Conf on Practice and Theory of Automated Timetabling: Springer
- [3] Schaerf, A. 1999, "A Survey of Automated Timetabling", Artificial Intelligence Review 13: 87-127, 1999. Netherlands: Kluwer Academic Publishers.
- [4] Abraham, A. *et al.* 2000, "Nature's Heuristics for Scheduling Jobs on Computational Grids", - Proceedings of the 8th IEEE Inter Conf on Adv Learning Technologies 2000:
- [5] Ersnt A.T. *et al* 2004, "An Annotated Bibliography of Personnel Scheduling and Rostering", - Annals of Operation Research 2004 Springer
- [6] Sims M. J. 1997 "An Introduction To Planning And Scheduling With Simulation", Proceedings of the 1997 Winter Simulation Conference ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson
- [7] Rivreau D. 2004, "Introduction to Scheduling Problems", - Institut de Mathematiques Appliquees. Université Catholique de l'Ouest. September 6, [online] Available from: www.uco.fr/~rivreau/Cours/sched-intro.pdf
- [8] Wolfe B. 2005, "Scheduling: Overview of Concepts and Methods", [online] Available from <http://faculty.csuci.edu/william.wolfe/csuci/Amgen/SchedulingPresentation.ppt> [Accessed 11 April 2006]
- [9] Burke E. K. *et al* 1997 - "Automated University Timetabling: The State of the Art", The Computer Journal 1997 40(9):565-571: British Computer Society
- [10] Filho G. R. and Lorena L. A. N. 2001 "A Constructive Evolutionary Approach to School Time Tabling", - Lecture Notes In Computer Science Vol. 2037 Proceedings of the EvoWorkshops on Applications of Evolutionary Computing 130 - 139
- [11] Chen S. F., Liu Y. 2001, "The Application of Multi-level Genetic Algorithms in Assembly Planning", - Journal of Industrial Technology, Vol 17-4, August, 2001
- [12] Cheng R, Gen M, Tsujimura Y, 1999 "A Tutorial Survey of Job Shop Scheduling Using Genetic Algorithms, part II: hybrid genetic search strategies", - Computers and Industrial Engineering 36 (1999) 343 - 364
- [13] Coello Coello C. A., 2000, "An Updated Survey of GA-Based Multiobjective Optimisation Techniques ACM", Computing Surveys, Vol. 32, No. 2, June 2000
- [14] Deb K. *et al.* 2002, "A Fast Elitism Non-Dominated Sorting Algorithm for Multiobjective Optimisation": Proceedings of the Parallel Problem Solving from Nature (VI) Conference. Lecture Notes in Computer Science No. 1917 Paris: Springer
- [15] Watson J., Whitley L.D. and Howe A. E. 2005, "Linking Search Space Structures, Run-Time Dynamics and

- Problem Difficulty: *A Step Towards Demystifying Tabu Search*", *Journal of Artificial Intelligence Research* 24 (2005) 221 – 261
- [16] Lee J. *et al.* 2005 "University Timetabling through Conceptual Modeling",
- [17] *International Journal of Intelligent systems*, vol. 20, 1137–1160 2005 Wiley Periodicals, Inc.
- [18] Locatelli M. 2000 "Simulated Annealing Algorithms for Continuous Global Optimisation", *Journal of Optimization Theory and Applications*, Vol. 104, No. 1, pp.121–133, January 2000 – Springer
- [19] Negnevitsky, N. 2005, "Artificial Intelligence: A Guide To Intelligent Systems", 2nd ed. London: Addison-Wesley
- [20] Ponnambalam S. G., Aravindan P., Rajesh S. V., 2000, "A Tabu Search Algorithm for Job Shop Scheduling", *Int J Adv Manuf Technol* (2000) 16:765–771, London: Springer-Verlag
- [21] Braune R., Wagner S., Affenzeller M., 2004, "Applying Genetic Algorithms to the Optimization of Production Planning in a Real-World Manufacturing Environment", – *Cybernetics and Systems* 2004.,
- [22] Wangtong *et al* 2002 "Comparing Three Heuristic Methods for Functional Partitioning Hardware – Software Codesign", *Design Automation for Embedded Systems*, 6, 425 – 449, 2002 Boston: Kluwer Academic Publishers
- [23] J. van Leeuwen 2003 "Introduction To Scheduling Problems", [online] Available From: <http://www.cs.uu.nl/docs/vakken/amc/lecture03-10.pdf> [Accessed 10 April 2006]
- [24] Aicklein U., 2002 "An Indirect Genetic Algorithm for Set Covering Problem", – *Journal of the Operational Research Society* (2002) 53, 1118 – 112
- [26] Fonseca C. M. and Fleming P. J., 1995, "An Overview of Evolutionary Algorithms in Multiobjective Optimisation", *Evolutionary Computation*, 3(1):1 – 16, 1995.
- [27] Vincent, J. 2003, "Numerical Optimisation Using Genetic Algorithms", [online] School of DEC, Bournemouth University. Available from: http://dec.bournemouth.ac.uk/staff/jvincent/teaching/cc/numerical_optimisation_using_genetic_algorithms.pdf [Accessed 08 April 2006].
- [28] Alba E., Laguna M., Luque G., 2005, "Workforce Planning with a Parallel Genetic Algorithm", In *CEDI-MAEB'05*, pages 911 – 919, Granada, España, 2005
- [29] Globus A, *et al* 2001 "Scheduling Earth Observing Satellites with Evolutionary Algorithms", [online] Available from: <http://alglobus.net/NASAwork/papers/SMCIT03/SMCIT02paper2.pdf> [Accessed 09 April 2006]
- [30] Alkan A, Ozean E. 2003 – "Memetic Algorithm for Time Tabling", – *IEEE Congress on Evolutionary Computation*, 2003
- [31] Coates, G. *et al.* 2002, "Real-Time Coordinated Scheduling Using A Genetic Algorithm", - *Proceedings of the 5th International Conference on Adaptive Computing in Design and Manufacture (ACDM '02)*, Exeter, UK, 381 – 392, 16 – 18 April 2002
- [32] Globus, A. *et al.* (2002) "Scheduling Earth Observing Fleets with Evolutionary Algorithms : Problem Description and Approach", *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, NASA, Houston, Texas, 27-29 October, 2002.

- [25] Wolfe, W. J. and Sorensen, S. E., 2000, "Three Scheduling Algorithms Applied to the Earth Observing Systems Domain", *Management Science* © 2000 INFORMS Vol. 46, No. 1, January 2000 pp. 148-168.
- [33] De Bontridder K J M 2005, "Minimizing Total Weighted Tardiness In A Generalized Job Shop", *Journal of Scheduling* 8: 479 - 496, 2005. Netherlands: Springer
- [34] Hertz A., Taillard E., de Werra D. 1992 "A Tutorial on Tabu Search",
- [35] Eiben A. E. and Schoenauer M. 2002 "Evolutionary Computing", - *Information Processing Letters* 82 (2002) 1-6: Elsevier Science
- [36] Wang L. and Zheng D. Z. "A Modified Genetic Algorithm for Job Shop Scheduling", *International Journal of Advanced Manufacturing Technology*, 2002, 20(1): 72-76
- [37] Whitley D. 1994, "A Genetic Algorithm Tutorial", - *Statistics and Computing* 4, 65-85
- [38] Weare R., Burke E, and Elliman D. 1995 "A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems", *Proceedings of the 6th International Conference on Genetic Algorithm* 1995 605 - 610.
- Proceedings of Giornate di Lavoro AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes)
- [39] Burke E. K., Kendall G, and Soubeiga E. 2004 "A Tabu Search Hyperheuristics for Time Tabling and Rostering", *Journal of Heuristics*, 9: 451-470, 2004 Netherlands, Kluwer Academic Publishers.
- [40] Han L, Kendall G, Cowling P, 2002 - "An Adaptive Length Chromosome Hyperheuristic Genetic Algorithm for a Trainer Scheduling Problem", - 4th Asia-Pacific Conference on Simulated Evolution and Learning
- [41] Wolpert D, Macready W, 1996. "No free lunch theorems for search," Santa Fe Institute, Technical report SFI-TR-95-02-010.
- [42] Burke E and Kendall G 1999, "Comparison of Meta-Heuristic Algorithms for Clustering Rectangles", *Computers And Industrial Engineering*, 37, Issue 1-2 (October 1999) 383 - 386 NY: Pergamon Press, Inc
- [43] Burke E.K. and Petrovic S. "Recent Research Directions in Automated Time Tabling", *European Journal of Operational Research-EJOR*, 140/2, 2002, 266-280 : Elsevier Science..