Journal of Heuristics, 6: 457–480 (2000)

New and "Stronger" Job-Shop Neighbourhoods: A Focus on the Method of Nowicki and Smutnicki (1996)

ANANT SINGH JAIN
i2 Technologies UK, The Priory, Stomp Road, Burnham, Buckinghamshire, England, UK, SL1 7LL
email: anant_jain@i2.com

BALASUBRAMANIAN RANGASWAMY
Optimisation and Logistics Modelling Group, US WEST Advanced Technologies, 4001 Discovery Drive, Boulder, CO 80303, USA
email: brangas@advtech.uswest.com

SHEIK MEERAN*
Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, UK, DD1 4HN

Abstract

Examination of the job-shop scheduling literature uncovers a striking trend. As methods for the deterministic job-shop problem have gradually improved over the years, they have come to rely on neighbourhoods for selecting moves that are more and more constrained. We document this phenomenon by focusing on the approach of Nowicki and Smutnicki (*Management Science*, 1996, 42(6), 797–813), noted for proposing and implementing the most restrictive neighbourhood in the literature. The Nowicki and Smutnicki (NS) method which exploits its neighbourhood by a tabu search strategy, is widely recognised as the most effective procedure for obtaining high quality solutions in a relatively short time. Accordingly, we analyse the contribution of the method's neighbourhood structure to its overall effectiveness. Our findings show, surprisingly, that the NS neighbourhood causes the method's choice of an initialisation procedure to have an important influence on the best solution the method is able to find. By contrast, the method's choice of a strategy to generate a critical path has a negligible influence. Empirical testing further discloses that over 99.7% of the moves chosen from this neighborhood (by the NS rules) are disimproving—regardless of the initial solution procedure or the critical path generation procedure employed. We discuss implications of these findings for developing new and more effective job-shop algorithms.

Key Words: neighborhood, tabu search, job-shop, scheduling

1. Introduction

A neighbourhood, $N(x)$ is a function which defines a simple transition from a solution x to another solution by inducing a change that typically may be viewed as a small perturbation (Glover and Laguna, 1997).[1] Each solution $x' \in N(x)$ can be reached directly from x by a

*Present address: School of Manufacturing and Mechanical Engineering, University of Birmingham, UK, B152 TT, UK. Email: s.meeran@bham.ac.uk

single predefined partial transformation of x called a move, and x is said to move to x' when such an transition is performed. (The term *solution* is conceived in the sense of one that satisfies certain structural requirements of a problem at hand, but it is not necessary that all requirements qualify as feasible.) In the majority of neighbourhood searches a symmetry is assumed to hold where x' is a neighbour of x if and only if x is a neighbour of x'. A neighbourhood function, more formally, is a mapping $N : \mathcal{R} \rightarrow 2^{\mathcal{R}}$ which associates a set of solutions $\mathcal{R}(x)$ with each solution $x \in \mathcal{R}$ obtained by a move (Glover and Laguna 1997). The objective of these strategies is to progressively perturb the current configuration through a succession of neighbours in order to direct the search to an improved solution. Improvement is sought at each step by standard *ascent* methods, or in some (possibly) larger number of steps by more advanced methods. In cases where solutions may involve infeasibilities, improvement is often defined relative to a modified objective that penalises such infeasibilites.

This work focuses on the application of neighbourhood search to the problem of job shop scheduling. The deterministic job shop (DJS) problem provides a useful domain for analysis because it is an important model in scheduling theory and is known to be extremely difficult to solve. It serves as a proving ground for new algorithmic ideas and is strongly motivated by practical requirements. In addition a great deal of previous research has been done that provides a basis for comparisons among alternative approaches, old and new.

1.1. The deterministic job shop problem

The DJS problem considered here is of the type given in French (1982), which consists of a finite set $\mathcal{J}$ of n jobs $\{\mathcal{J}_i\}_{i=1}^{n}$ to be processed on a finite set $\mathcal{M}$ of m machines $\{\mathcal{M}_k\}_{k=1}^{m}$. Each job $\mathcal{J}_i$ must be processed on every machine and consists of a chain or complex of m_i operations $O_{i1}, O_{i2}, \ldots, O_{im_i}$, which have to be scheduled in a predetermined order, a requirement called a *precedence constraint*. There are N operations in total, $N = \sum_{i=1}^{n} m_i \cdot O_{ik}$ is the operation of job $\mathcal{J}_i$ which has to be processed on machine $\mathcal{M}_k$ for an uninterrupted processing time period τ_{ik} and no operation may be pre-empted, i.e., interrupted and then completed at a later date. (However a simpler notation is used in this paper by allocating each operation with a unique number from 1 to (nm). The following scheme is applied in this indexing of operations. The first operation of $\mathcal{J}_1$ is 1, the first operation of $\mathcal{J}_2$ is 2, ... the first operation of $\mathcal{J}_n$ is n, the second operation of $\mathcal{J}_1$ is $(n+1), \ldots$ and the last operation of $\mathcal{J}_n$ is (nm). More formally if $\mathcal{M}_k$ is the kth machine required by O_{ik}, with respect to precedence constraints, then its operation number is $n(k-1)+i$ (c.f. figure 1).) Each job has its own individual flow pattern through the machines which is independent of the other jobs. The problem is defined by *capacity constraints* which stipulate that each machine can process only one job and each job can be processed by only one machine at a time. The duration in which all operations for all jobs are completed is referred to as the makespan $C_{\max}$. The objective of the scheduler is to determine starting times for each operation, $t_{ik} \geq 0$, in order to minimise the makespan while satisfying all the precedence and capacity constraints. That is, the goal maybe expressed as determining $C^*_{\max}$, where

$$C^*_{\max} = \min(C_{\max}) = \min_{\text{feasible schedules}} (\max(t_{ik} + \tau_{ik}) : \forall \mathcal{J}_i \in \mathcal{J},\ \mathcal{M}_k \in \mathcal{M}).$$