Improving Neighborhoods for Local Search Heuristics

Part 1

Peter Brucker
Universität Osnabrück
Johann Hurink
Universität Osnabrück
Frank Werner
Technische Universität Magdeburg

Extended Abstract

Local search techniques are useful tools for solving discrete optimization problems. Popular heuristics like simulated annealing and tabu search are based on such techniques. These methods depend on an underlying neighborhood structure. Usually, the quality of the neighborhood structure has some important influence on the methods.

Our approach is to improve methods by replacing neighborhood structures by secondary neighborhood structures derived form the original ones. We apply these ideas to some NP-hard scheduling problems.

A **discrete optimization** problem can be described as follows. For a given finite set S and a given function $c : S \to \mathbb{R}$ one has to find a solution $s^* \in S$ with

$$c(s^*) \leq c(s) \text{ for all } s \in S.$$

S is called **feasible set**.

Local search is an iterative procedure which moves from one solution $s \in S$ to another solution repeating this step as long as it seems to be necessary. The possible moves from s to the next solutions are restricted by a set $OP(s)$ of possible operators $op : S \to S$. Thus,

$$\mathcal{N}(s) := \{op(s) \mid op \in OP(s)\}$$

is the set of all possible **neighbors** of s. The operator sets $OP(s), s \in S$ define a neighborhood structure

$$\mathcal{N}(s) \quad s \in S$$

on the set S.

The simplest local search method is the method of **iterative improvement** which chooses the best solution in $\mathcal{N}(s)$ as the solution to move to from s. It stops if no solution in $\mathcal{N}(s)$

improves the solution s. In this case s is a **local optimum**. Unfortunately the value $c(s)$ of a local optimum may be far away from the optimal value. To avoid this problem simulated annealing and tabu search allow moves to nonimproving solutions. Still a disadvantage of these methods is an ozillation around local optima which results in a slow convergence.

Our approach to overcome these difficulties is to replace the original feasible set S_1 by the subset S_2 of all $s \in S_1$ which are locally optimal with respect to a neighborhood structure $\mathcal{N}_1(s), s \in S_1$ on the set S_1. Furthermore, we construct new operator sets $OP_2(s), s \in S_2$ which define a new neighborhood structure $\mathcal{N}_2(s), s \in S_2$ on the set S_2.

The advantages of such an approach are

- the search space is reduced considerably
- local search methods can be still applied (at a higher level)
- ozillations, appear only at a higher level.

Furthermore, the construction of the operator sets $OP_2(s)$ is problem specific, i.e. structural properties are taken into consideration. A disadvantage is that the method is not a general purpose method. Different problems have to be treated differently.

A neighborhood structure is **connected** if for any two feasible solutions s_1 and s_2 solution s_2 is reachable from solution s_1 by a sequence of moves. Experiments have shown that in general a good neighborhood structure must be connected. All neighborhood structures we are considering are connected.

We apply this approach to the following NP-hard scheduling problems with corresponding neighborhood structures.

(a) $P2 \mid\mid C_{\max}$

$P2 \mid\mid C_{\max}$ denotes the problem of scheduling n jobs $i = 1, \ldots, n$ with processing times $p_i (i = 1, \ldots, n)$ on two identical parallel machines such that the makespan is minimized. A feasible solution of this scheduling problem is given by a partitioning of the job set $I = \{1, \ldots, n\}$ into two disjoint sets I_1 and I_2. We denote such a partitioning by (I_1, I_2). I_ν is the set of jobs to be processed on machine $M_\nu (\nu = 1, 2)$. For $\nu = 1, 2$ let $s_\nu := \sum_{i \in I_\nu} p_i$ be the total processing time on machine M_ν. Then $\max\{s_1, s_2\}$ is the makespan of the schedule defined by (I_1, I_2). We have to find a partitioning (I_1, I_2) such that

$$\max\{s_1, s_2\} = \max\left\{\sum_{i \in I_1} p_i, \sum_{i \in I_2} p_i\right\}$$

is minimized. The problem is show to be NP-hard by a simple reductions from the partitioning problem.

For this problem a neighborhood N_1 is defined by the set S_1 of all feasible solutions (I_1, I_2) and the operators move (i) $(i = 1, \ldots, n)$. move (i) moves job i from the machine on which i is scheduled to the other machine, i.e.

$$move(i)(I_1, I_2) = \begin{cases} (I_1 \backslash \{i\}, I_2 \cup \{i\}) & \text{if } i \in I_1 \\ (I_1 \cup \{i\}, I_2 \backslash \{i\}) & \text{if } i \in I_2. \end{cases}$$

(b) $1 \mid prec \mid \sum C_i$

$1 \mid prec \mid \sum C_i$ denotes the problem of scheduling n jobs $1, ..., n$ with processing times p_i $(i = 1, ..., n)$ on one machine such that the mean flow time is minimized. Between the jobs precedence relations $\rightarrow$ are given (a precedence relation $i \rightarrow j$ expresses that i has to be processed before j). A feasible schedule for this problem is given by a feasible sequence (permutation) $\pi = (\pi_1, ..., \pi_n)$ of the jobs which is compatible with the precedence relation. The corresponding completion times for the job π_i is given by $C_i = \sum_{j=1}^{i} p_{\pi_j}$, $(i = 1, ..., n)$. We have to find a feasible sequence π such that

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{n} \sum_{j=1}^{i} p_{\pi_j}$$

is minimized.

For this problem a neighborhood is defined by the set S_1 of all feasible sequences π and by the operators $exchange(i)$, $(i = 1, ..., n)$. The operator $exchange(i)$ will exchange the jobs which are scheduled in position i and $i+1$, i.e the sequence $\pi' = exchange(i)(\pi)$ is defined by

$$\pi'_k = \begin{cases} \pi_{i+1} & \text{if } k = i \\ \pi_i & \text{if } k = i+1 \\ \pi_k & \text{else} \end{cases}$$

The operator $exchange(i)$ maps a feasible schedule $\pi \in S_1$ into a feasible schedule iff there does not exist a precedence relation $\pi_i \rightarrow \pi_{i+1}$.

We define the neighborhood $N_1(\pi)$ by the set of operators

$$OP_1(\pi) = \{exchange(i) \mid i = 1, ..., n-1;\ \pi_i \rightarrow \pi_{i+1} \text{ is not a precedence relation}\}$$

(c) $1 \parallel \sum |T_i$

$1 \parallel \sum |T_i$ denotes the problem of scheduling n jobs $1, \ldots, n$ on a single machine such that the total tardiness $\sum T_i$ is minimized. $T_i = \max\{0, C_i - d\}$ is the tardiness of job i and C_i denotes its completion time. We assume that $d_1 \leq d_2 \leq \ldots \leq d_n$. Then the following lemma holds.

Lemma 1: Let i, k be two jobs with $p_i \leq p_k$ and $i < k$. Then there exists an optimal solutions where job i is processed before job k.

Hence, we can establish a precedence constraint $i \rightarrow k$ between job i and k if the condition of Lemma 1 holds. We need only consider the set S_1 of sequences which are compatible with all constraints established by Lemma 1.

For the problem $1 \parallel \sum |T_i$ the neighborhood is now defined on the set of the feasible sequences S_1 in the same way as in the previous section.

For each neighborhood defined in connection with problems (a), (b) and (c) we will

- characterize the local optima
- define a new neighborhood on the set of the local optima
- prove that the new neighborhood is connected.

More details will be discussed in Part 2 of this paper.