# A new inter-island genetic operator for optimization problems with block properties

Wojciech Bożejko[1] and Mieczysław Wodecki[2]

[1] Institute of Engineering Cybernetics, Wrocław University of Technology
Janiszewskiego 11-17, 50-372 Wrocław, Poland
email: wbo@ict.pwr.wroc.pl
[2] Institute of Computer Science, University of Wrocław
Przesmyckiego 20, 51-151 Wrocław, Poland
email: mwd@ii.uni.wroc.pl

**Abstract.** Combinatorial optimization problems of scheduling belongs in most cases to the NP-hard class. In this paper we propose very effective method of construct parallel algorithms based on island model of coevolutionary algorithm. We apply block properties, which enable inter-island genetic operator to distribute calculations and shorten communication between processors. *

## 1 Introduction

Many methods of algorithms construction consist in looking through (directly or indirectly) all or a part of set of feasible solutions. Such a mechanism is based on generating from current (base) solution next solution, or a set of solutions (so called neighborhood), from which the best solution is chosen. This solution is the base solution in the next iteration. Such a mechanism can be met (among others) in Branch and Bound (B&B) method and in many other algorithms which consist of improving the solution, as well as in the best (nowadays) approximate algorithms, metaheuristics: tabu search and simulated annealing methods. The same method is used in the path-relinking method, which is used in the local search genetic operators, such as Multi – Step Crossover Fusion (MSXF) of Reeves and Yamada [9]. Quality of these algorithm's solutions depends on: number of iteration, the method of neighborhood describing and its reviewing. The time of computations can be shorten by its realization in multiprocessor environment. Unfortunately parallelization of the sequential algorithms directly (for example by using parallel compiler) does not give satisfactory speedup. In this paper we propose some new elements of parallel local search algorithms. Partitioning solution (permutation) into blocks (subpermutations) enables to decrease neighborhood size and its division into separated subsets. It makes possible its generating and reviewing independently on parallel machine. We also use parallel computer for executing coevolutionary genetic algorithm with inter-island genetic operator based on local search procedure with blocks.

* The work was supported by KBN Poland, within the grant No. 4T11A01624

## 2 Blocks in solutions

For the *TWET-no-idle* problem, each schedule of jobs can be represented by permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ of the set of jobs $J$. Let $S(n)$ denote the set of all such permutations. The total cost $\pi \in S(n)$ is $F(\pi) = \sum_{i=1}^{n} f_{\pi(i)}(C_{\pi(i)})$, where $C_{\pi(i)}$ is completed time of the job $\pi(i)$, $C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j)}$. The job $\pi(i)$ is considered *early* if it is completed before its earliest moment of finishing ($C_{\pi(i)} < e_{\pi(i)}$), *on time* if $e_{\pi(i)} \leq C_{\pi(i)} \leq d_{\pi(i)}$, and *tardy* if the job is completed after its due date (i.e. $C_{\pi(i)} > d_{\pi(i)}$).

Each permutation $\pi \in S(n)$ is decomposed into subpermutations (subsequences of jobs) $\Omega = [B_1, B_2, \ldots, B_v]$ called *blocks* in $\pi$, each of them contains the jobs, where:

1. $B_i = (\pi(a_i), \pi(a_i + 1), \ldots, \pi(b_i - 1), \pi(b_i))$, and
   $b_i = a_{i-1} + 1, \ \ 1 \leq i \leq v, \ \ a_0 = 0, \ b_v = n.$
2. All the jobs $j \in B_i$ satisfy the following condition:
   $e_j > C_{\pi(b_i)}$, or (C1)
   $e_j \leq C_{\pi(b_{i-1})} + p_j$ and $d_j \geq C_{\pi(b_i)}$, or (C2)
   $d_j < C_{\pi(b_{i-1})} + p_j$. (C3)
3. $B_i$ are maximal subsequences of $\pi$ in which all the jobs satisfy either Condition $C1$ or Condition $C2$ or Condition $C3$.

By definition, there exist three types of blocks implied by either C1 or C2 or C3. To distinguish them, we will use the *E-block*, *O-block* and *T-block* notions respectively. For any block $B$ in a partition $\Omega$ of permutation $\pi \in S(n)$, let

$$F_B(\pi) = \sum_{i \in B} (u_i E_i + w_i T_i).$$

Therefore, the value of a goal function

$$F(\pi) = \sum_{i=1}^{n} (u_i E_i + w_i T_i) = \sum_{B \in \Omega} F_B(\pi).$$

If $B$ is a $T$-block, then every job inside is early. Therefore, an optimal sequence of the jobs within $B$ of the permutation $\pi$ (that is minimizing $F_B(\pi)$) can be obtained, using the well-known Weighted Shortest Processing Time ($WSPT$) rule, proposed by Smith [8]. The $WSPT$ rule creates an optimal sequence of jobs in the non-increasing order of the ratios $w_j/p_j$. Similarly, if $B$ is an $E$-block, than an optimal sequence of the jobs within can be obtained, using the Weighted Longest Processing Time ($WLPT$) rule which creates a sequence of jobs in the non-decreasing order of the ratios $u_j/p_j$.

Partition $\Omega$ of the permutation $\pi$ is *ordered*, if there are jobs in the $WSPT$ sequence in any $T$-block, and there are jobs in the $WLPT$ sequence in any $E$-block.

**Theorem 1** [2] Let $\Omega$ be an ordered partition of a permutation $\pi \in S(n)$ to blocks. If $\beta \in S(n)$ and $F(\beta) < F(\pi)$, so at least one job of some block of $\pi$ was moved before the first or after the last job of this block in permutation $\beta$.

Note that Theorem 1 provides the necessary condition to obtain a permutation $\beta$ from $\pi$ such, that $F(\beta) < F(\pi)$.

Let $\Omega = [B_1, B_2, \ldots, B_v]$ be an ordered partition of the permutation $\pi \in S(n)$ to blocks. If a job $\pi(j) \in B_i$ ($B_i \in \Omega$), therefore moves which can improving goal function value consists in reordering a job $\pi(j)$ before the first or after the last job of this block. Let $N_j^{bf}$ and $N_j^{af}$ be sets of such moves ($N_j^{bf} = \emptyset$ for $j \in B_1$ and $N_j^{af} = \emptyset$ for $j \in B_v$). Therefore, the neighborhood of the permutation $\pi \in S(n)$,

$$N(\pi) = \bigcup_{j=1}^{n} N_j^{bf} \cup \bigcup_{j=1}^{n} N_j^{af}. \tag{1}$$

There are many other problems with blocking partitioning property, i.e.

1. Flow shop problem – subsets of jobs from the same machine on critical path are blocks, see. Nowicki and Smutnicki [6], Grabowski and Pempera [3], Grabowski and Wodecki [4].
2. Job shop problem – subsets of jobs from the same machine on critical path are blocks, see. Nowicki and Smutnicki [7], Grabowski and Wodecki [5] .
3. Classic single machine total weighted tardiness problem (TWTP) – subsets of jobs made on time and made after deadline are blocks, it is special case of the TWET-no-idle problem.

Methods proposed here can be directly applied to all of these problems.

## 3 Parallel genetic algorithm

The island model of parallel genetic algorithm is characterized by a significant reduction of the communication time, compared to global model (with distributed computations of the fitness function only). Shared memory is not required, so this model is more flexible too.

Below, a parallel genetic algorithm is proposed. There is the MSXF (Multi – Step Crossover Fusion) operator used to extend the process of researching for better solutions of the problem. MSXF has been described by Reeves and Yamada [9]. Its idea is based on local search, starting from one of the parent solutions, to find a new good solution where the other parent is used as a reference point. Additionaly, block properties was used to make a search process more effective – to prevent changes inside the block, which are nieopacalne from the fitness function's point of view. Such a postpowanie odpowiada an idea of not making changes between genes of different chromosomes. In such a way a MSXF+B (MSXF with blocks) operator was created.

The neighborhood $N(\pi)$ of the permutation (individual) $\pi$ is defined as a set of new permutations that can be reached from $\pi$ by exactly one adjacent pairwise exchange operator which exchanges the positions of two adjacent jobs of a problem's solution connected with permutation $\pi$. The distance measure $d(\pi,\sigma)$ is defined as a number of adjacent pairwise exchanges needed to transform permutation $\pi$ into permutation $\sigma$. Such a measure is known as Kendall's $\tau$ measure.

**Algorithm 1.** Multi-Step Crossover Fusion with blocks (MSXF+B)

Let $\pi_1$, $\pi_2$ be parent solutions. Set $x = q = \pi_1$;
**repeat**
  For each member $y_i \in N(\pi)$, calculate $d(y_i, \pi_2)$;
  Sort $y_i \in N(\pi)$ in ascending order of $d(y_i, \pi_2)$;
  **repeat**
    Select $y_i$ from $N(\pi)$ with a probability inversely
      proportional to the index $i$; Calculate $C_{sum}(y_i)$;
    Accept $y_i$with probability 1 if $C_{sum}(y_i) \leq C_{sum}(x)$, and with
      probability $P_T(y_i) = exp((C_{sum}(x) - C_{sum}(y_i)) \ / \ T)$ otherwise
      ($T$ is temperature);
    Change the index of $y_i$ from $i$ to $n$ and the indices of
      $y_k$, $k = i{+}1,...,n$ from $k$ to $k{-}1$;
  **until** $y_i$ is accepted;
  $x \leftarrow y_i$; **if** $C_{sum}(x) < C_{sum}(q)$ **then** $q \leftarrow x$;
**until** *some termination condition is satisfied*;
$q$ is the offspring.

In our implementation, MSXF+B is an inter-island (i.e. inter-subpopulation) crossover operator which constructs a new individual using the best individuals of different islands connected with subpopulations on different processors. The condition of termination consisted in exceeding of 100 iterations by the MSXF+B function.

**Algorithm 2. Parallel genetic algorithm**

**parfor** $j = 1, 2, ..., p$ { *$p$ is number of processors* }
  $i \leftarrow 0$; $P_j \leftarrow$ random subpopulation connected with processor $j$;
  $p_j \leftarrow$ number of individuals in $j$ subpopulation;
  **repeat**
    Selection$(P_j, P'_j)$; Crossover$(P'_j, P''_j)$; Mutation$(P''_j)$;
    **if** ($k$ mod $R = 0$) **then** {*every $R$ iteration*}
      $r :=$ *random*(1, $p$); MSXF+B$(P'_j(1), P_r(1))$;
    **end if;**
    $P_j \leftarrow P''_j$; $i \leftarrow i + 1$;
    **if** *there is no improvement of the average* $C_{sum}$ **then** {*Partial restart*}
      $r :=$ *random*(1,$p$);
      Remove $\alpha = 90$ percentage of individuals in subpopulation $P_{j.}$;
      Replenish $P_j$ by random individuals;
    **end if**;
    **if** ($k$ mod $S = 0$) **then** {*Migration*}

$r := random(1,p)$;
Remove $\beta = 20$ percentage of individuals in subpopulation $P_j$;
Replenish $P_j$ by the best individuals from subpopulation $P_r$
taken from processor $r$;
**end if;**
**until** *Stop_Condition*;
**end parfor**

The frequency of communication between processors (MSXF+B operator and migration) is very important for the parallel algorithm performance. It must not be too frequent because of the relative long time of communication between processors, comparing to the time of communication inside the program of a one processor. In this implementation the processor gets new individuals quite rarely, every $R = 20$ (MSXF+B operator) or every $S = 35$ (migration) iterations.

## 4 Computer simulations

The algorithm was implemented in the Ada95 language and run on 4-processors Sun Enterprise 4 x 400 MHz under the Solaris 7 operating system. Tasks of the Ada95 language were executed in parallel as system threads. Tests were based on 125 instances with 40,50 and 100 jobs taken from the OR-Library [8]. The results were compared to the best known, also taken from [8].

**Table 1.** Relative deviation of solutions of sequence and parallel genetic algorithms compared to the best known solutions.

| $n$ | 1 processor | | 4 processors | |
|---|---|---|---|---|
| | aver. dev. | max. dev. | aver. dev. | max. dev. |
| 40 | 2.907 | 99.963 | 0.057 | 1.534 |
| 50 | 4.035 | 167.576 | 0.064 | 1.362 |
| 100 | 0.005 | 1.054 | 0.004 | 0.103 |
| **average** | **2.317** | **89.531** | **0.042** | **0.999** |

The computational results can be found in Table 1. The number of iterations was counted as a sum of iterations on processors, and was permanently set to 800. For example, 4-processor implementations make 200 iterations on each of the 4 processors, so we can obtain comparable costs of computations. As we can see, parallel versions of the algorithm has much better results of the average and maximal relative deviation to the optimal (or the best known) solutions, working (parallel) in a shorter time. Because of small cost of the communication, the speedup parameter of the parallel algorithms is almost linear.

## 5 Conclusions

We have discussed a new approach to optimization problems with block properties based on the new inter-island genetic operator for the parallel asynchronous coevolutionary algorithm. Compared to the sequential algorithm, parallelization shorts computation's time and improving quality of the obtained solutions. The advantage of the parallel algorithm is especially visible for large instances of the problem.

## References


1. Bożejko W., Wodecki M., Parallel genetic algorithm for minimizing total weighted completion time, Lecture Notes in Computer Science No. **3070**, Springer Verlag 2004, 400–405.
2. Bożejko W., Wodecki M., Parallel genetic algorithm for the flow shop scheduling problem, Lecture Notes in Computer Science No. **3019**, Springer Verlag 2004, 566–571.
3. Grabowski J., Pempera J., New block properties for the permutation flow-shop problem with application in TS, Journal of the Operational Research Society, **52** (2001), 210–220.
4. Grabowski J., Wodecki M., A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion, Computers and Operations Research, **31** (2004), 1891–1909.
5. Grabowski J., Wodecki M., A very fast tabu search algorithm for job shop problem, in: Rego C., Alidaee B. (editors), Adaptive memory and evolution: tabu search and scatter search, Kluwer Academic Publishers, Dordrecht, 2004.
6. Nowicki E., Smutnicki C., A fast tabu search algorithm for the permutation flow shop problem, European Journal of Operational Research, **91** (1996), 160–175.
7. Nowicki E., Smutnicki C., A fast tabu search algorithm for the job shop problem, Management Science, **42**, 6 (1996), 797–813.
8. OR-Library: http://people.brunel.ac.uk/ mastjjb/jeb/info.html
9. Reeves C. R., Yamada T., Solving the Csum Permutation Flowshop Scheduling Problem by Genetic Local Search, IEEE International Conference on Evolutionary Computation (1998), 230–234.