JEDNA STRATEGIJA ZA PARALELIZACIJU VNS METODE ZA REŠAVANJE PROBLEMA RASPOREDJIVANJA*

A STRATEGY FOR PARALLELIZATION OF VNS FOR MULTIPROCESSOR SCHEDULING PROBLEM

Tatjana Davidović[1], Teodor Gabriel Crainic[2]
[1]MATEMATIČKI INSTITUT SANU, Knez-Mihailova 35/1, 11001 Beograd
[2]DEPARTEMENT MANAGEMENT ET TECHNOLOGIE,
Université du Québec à Montréal and
Centre de recherche sur les transports, Université de Montréal

Sažetak: *U radu se predlaže jedna od mogućih strategija paralelizacije metode promenljivih okolina za primenu na problem rasporedjivanja. Cilj paralelizacije je kako ubrzanje izvršavanja korišćenjem više procesora, tako i popravljanje kvaliteta dobijenog rešenja obezbedjivanjem pretraživanja na širem regionu prostora rešenja.*

KLJUČNE REČI: METODA PROMENLJIVIH OKOLINA, PROBLEM RASPOREDJIVANJA, PARALELIZACIJA, UBRZANJE IZVRŠAVANJA.

Abstract: *In this paper we propose a possible strategiz for the parallelization of the VNS heuristics to solve multiprocessor scheduling problem with communication delays.Our main goal is to speedup the executions by engaging several processors. Moreover, we hope to improve the quality by spreading up the search towards the different regions of the solution space.*

KEY WORDS: VARIABLE NEIGHBORHOOD SEARCH, MULTIPROCESSOR SCHEDULING PROBLEM, PARALLELIZATION, SPEEDUP OF EXECUTIONS.

1. INTRODUCTION

The Multiprocessor Scheduling Problem With Communication Delays (MSPCD) is as follows [4,5]: tasks (or jobs) have to be executed on several processors; we have to find where and when each task will be executed, such that the total completion time is minimum. A duration of each task is known as well as precedence relations among tasks, i.e. what tasks should be completed before some other could begin. In addition, if dependent tasks are executed on different processors, the data transferring time (or communication delay) that are given in advance are also considered. The tasks to be scheduled are represented by a weighted directed acyclic graph (DAG) [2,9], while multiprocessor architecture is assumed to contain identical processors and is modelled by a distance matrix [2,6].

*This research has been supported in part by NSF Serbia. The first and the third authors have been supported by grant no. 1583.

The sequential implementation of the VNS heuristic for solving MSPCD was poposed in [4]. Although performing best in average, VNS shows significant deviations in the cases with known optimal solutions. As one of the possible strategies for the performance improvement we propose its parallelization. There are different strategies for the parallelization of VNS heuristic. For example, the partitioning of the search space which may be realized in two ways: at low level by parallelization of LS procedure and at coarse level where in parallel we execute the whole VNS procedures on different parts of solution space. The other strategy may be the cooperation: several different (sequential or parallel) VNS methods run simultaneously and exchange the relevant information. In this paper we propose parallelization obtained by simultaneous execution of different steps of the VNS procedure. The paper is organized as follows: in the next section the original implementation of VNS heuristic is briefly described. Section 3 contains the description of parallelization strategy and experimental results on random test examples with known optimal solutions are given in section 4. Section 5 concludes the paper.

2. SEQUENTIAL VNS HEURISTIC FOR SCHEDULING PROBLEM

In applying VNS method to the MSPCD [4] we defined S, the set of all permutations of n tasks to be a set of solutions, and $X \subseteq S$ the set of feasible solutions (feasible permutation means that the order of tasks in that permutation obeys precedence constraints defined by the task graph: a task cannot appear before any of its predecessors or after any of its successors in a feasible permutation). Having a feasible permutation x, we are able to evaluate the objective function value in a unique way, if we follow always the same rule of assigning tasks to processors (for example, ES rule) in the order given by that permutation. Neighborhoods are defined using elementary transformations of permutation. We used here *Swap*-1 neighborhood. Next step in the implementation of VNS heuristic [4] is to define Local Search (LS) and Shaking procedures. LS procedure in given neighboirhood of a solution x consists of generation of all feasible neighbors and performing scheduling rule (ES) to calculate the value of objective function (schedule length). For shaking only *Swap*-1 neighborhood was used. Shaking procedure is defined by random selection of a task (or a pair of tasks) and perform the corresponding transformation.

3. PARALLELIZATION STRATEGY

The experiments performed with sequential implementation of permutation-based VNS [4,5] showed that for sparse task graphs, search space (set of feasible permutations) is too large to be exploited efficiently. The main goal of parallelization is to speedup the computations by engaging several processors and divide the total amount of work between them. This goal may be defined in one of the following two ways: 1) accelerate the search for the same quality solution or 2) improve the solution quality by allowing more processors to run the same amount of CPU time as the single one does. When metaheuristics are parallelized both definitions may be used simultaneously, i.e. the benefit may be doubled. Moreover, the third goal may be stated according to the nondeterminism of the metaheuristic search procedure: To explore distinct search regions and to further improve the solution quality.

The place suitable for parallelization is combination of shake and LS steps. They always go together and can be performed simultaneously in different ways. For example in multistart fashion, as it was implemented in [8], or in a way to allow the wider exploration of the search step, like in [1]. This last strategy for the parallelization of the VNS algorithm we implemented and called parallel VNS (PVNS). The main idea is to explore the different neighborhoods in parallel. This is realized by

performing shaking (in different neighborhoods) and LS (sequential) on different processors at the same time (Fig. 1).

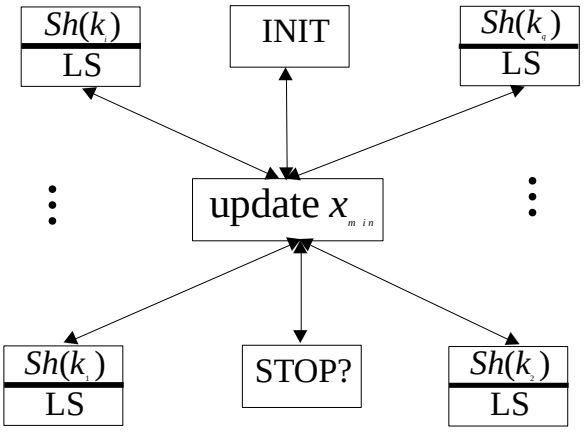

Fig. 1 Ilustration of the parallel VNS

We implemented this strateqy on multiprocessor system containing q+1 processors: one for the communication with user, initialization of the relevan data, coordination and synchronization between the other processors by collecting all solutions and updating global best one, taking care of the neighbourhood index update and the stopping criterion. Other processors perform the combination of shaking and LS procedure in given neighborhood.

4. EXPERIMENTAL EVALUATIONS

For the experiments with PVNS we use same examples as the ones used in the paper on sequential VNS [5]. The examples are generated according to the procedure described in [3], and have preselected values for the optimal solutions. We use ten examples of task graphs containing n=200 tasks with different edge densities and known value of the optimal schedule SL_{opt}=1200.

We tested several variants of the original (sequential) VNS algorithm and best results were obtained by using restricted version, the one that does not explore the whole neighborhood. In the table to follow, we give the results of the execution parallel communication-based restricted VNS (PCRVNS) on several multiprocessor architectures.

Table 1. Percentage of deviations from the known optimal solutions for n=200 within 10 repetitions

q	%dev		
	Single	av. Over 10	best of 10
1	22.16	22.16	22.16
5	18.16	20.71	17.54
10	16.47	19.87	16.32
15	15.40	18.23	13.57
20	15.27	16.95	10.24

We fix the other relevant VNS parameters and execute First Improvement (FI) LS procedure within given CPU time limit t_{max}=750 *sec*. The values for VNS parameters are [4]: k_{max}=100(n/2), k_{step}=1, *plateaux*=0.0.

As we can see from this table, the improvement with parallel VNS is significant although, we do not gain much with adding new processors since the search is becoming too random. To examine the speed of the parallel search we were following the improvements in time of the current best sollution and noticed the linear speedup for the execution: the largest improvements occur at the biginning of the execution.

5. CONCLUSION

In this paper we propose the parallelization of VNS heuristic for the multiprocessor scheduling problem with communication delays. The benefits were twofolds: we achieved linear speedup of the executions and improvement in the solution quality.

6. REFERENCES

[1] *Crainic T.G., Gendreau M., Hansen P., Mladenović N.,* (2004.) COOPERATIVE PARALLEL VARIABLE NEIGHBORHOOD SEARCH FOR THE *p*-MEDIAN, J. Heur. 10, 2003. (pp. 289-310)

[2] *Davidović T.,* (2000.) EXAUSTIVE LIST-SCHEDULING HEURISTIC FOR DENSE TASK GRAPHS. YUJOR, 10(1) (pp. 123-136).

[3] *Davidović T., Crainic T. G.,* (2003.) NEW BENCHMARKS FOR STATIC TASK SCHEDULING ON HOMOGENEOUS MULTIPROCESSOR SYSTEMS WITH COMMUNICATION DELAYS. Centre de Recherche sur les Transports, Technical report, CRT-2003-04, Montreal, Canada.

[4] *Davidović T., Hansen P., Mladenović N.,* (2004.) PERMUTATION-BASED GENETIC, TABU AND VARIABLE NEIGHBORHOOD SEARCH HEURISTICS FOR MULTIPROCESSOR SCHEDULING WITH COMMUNICATION DELAYS, GERAD Technical report, G-2004-19, Montreal, Canada.

[5] *Davidović T., Hansen P., Mladenović N.,* (2001.) SCHEDULING BY VNS: EXPERIMENTAL ANALYSIS. *SYM-OP-IS 2001,* Beograd, 2001. (pp. 319-322)

[6] *Djordjević G., Tošić M.,* (1996.) A COMPILE-TIME SCHEDULING HEURISTIC FOR MULTI-PROCESSOR ARCHITECTURES. The Computer Journal, 39(8), 1996. (pp. 663-674)

[7] *Mladenović N., Hansen P.,* (1997.) VARIABLE NEIGHBORHOOD SEARCH. Comp. Oper. Res., 24(11), 1997., (pp. 1097-1100)

[8] *García-López F., Melián-Batista B., Moreno-Pérez J.A., Moreno-Vega J.M.* (2002.), THE PARALLEL VARIABLE NEIGHBORHOOD SEARCH FOR THE *p*-MEDIAN PROBLEM. J. Heur., 8(3), May 2002. (pp. 375-388)

[9] *Sih G. C., Lee E. A.,* (1993.) A COMPILE-TIME SCHEDULING HEURISTIC FOR INTERCONNECTION-CONSTRAINED HETEROGENEOUS PROCESSOR ARCHITECTURES. IEEE Trans. on Paral. Dist. Syst., 4(2), Feb. 1993. (pp. 175-187)