RASPOREDJIVANJE METODOM PROMENLJIVIH OKOLINA: EKSPERIMENTALNA ANALIZA

SCHEDULING BY VNS: EXPERIMENTAL ANALYSIS*

Tatjana Davidović
[1]Matematički institut SANU, Beograd, p.f. 367. email:tanjad@mi.sanu.ac.yu
Pierre Hansen
[2]GERAD and Ecole des Hautes Etudes Commerciales,3000 chemin de la Cote-Sainte-Catherine,
Montreal H3T 2A7, Canada email: pierreh@crt.umontreal.ca
Nenad Mladenović [1,2]
email: nenad@mi.sanu.ac.yu, nenad@crt.umontreal.ca

Rezime – *U ranijim radovima razvijali smo i uporedjivali razne heuristike za rasporedjivanje zadataka u prisustvu komunikacija i zaključili da metoda promenljivih okolina u proseku daje najbolje rezultate za sve tipove slučajno generisanih grafova zadataka. Medjutim, pronašli smo klasu grafova sa poznatim optimalnim rešenjem za koju se heuristička raspodela dobijena metodom promenljivih okolina (iako je još uvek bolja od drugih) razlikuje od optimalne za više od 50%. U ovom radu opisaćemo analize koje smo izvršili sa ciljem razumevanja dobijenog odstupanja heurističkih rešenja od optimalnog. Kao rezultat dobili smo odstupanje samo 6%.*

KLJUČNE REČI: GRAF ZADATAKA, RASPOREDJIVANJE, KOMUNIKACIONO KAŠNJENJE, METODA PROMENLJIVIH OKOLINA.

Abstract - *In our previous papers we developed and compared different scheduling heuristics and concluded that VNS performs best in average for all types of random task graphs we generated. Yet, there is a class of task graphs with known optimal solutions such that VNS obtained schedule (although still the best one) differs from the optimal one for more than 50%. In this paper we describe the analysis we performed in order to explain the deviation of the heuristic solution from the optimal one. As a result, we are able to reduce the error to 6% above the optimal solution.*

KEY WORDS: TASK GRAPH, SCHEDULING, COMMUNICATION DELAY, VARIABLE NEIGHBORHOOD SEARCH.

1. INTRODUCTION

Metaheuristics such as Genetic Algorithms (GA), Variable Neighborhood Search (VNS), Tabu Search (TS) are very efficient and therefore very popular methods in solving combinatorial optimization problems.

Metaheuristics have already been applied to different variants of scheduling problems. In [9] TS was applied to problem of scheduling independent tasks. TS for scheduling communicationg tasks with significant communication delays onto heterogeneous multiprocessor architecture was developed in [7]. GA approach was developed for scheduling dependent tasks with neglected communication time in [1]. Even parallel variants of TS [8] and GA [5] have appeared. Acording to problem considered and/or solution representation it is very hard to compare these existing metaheuristic approaches. We developed VNS, TS [2] and GA [3] for solving MSPCD based on same solution representation and compared them onto random task graphs with given density and sometimes with given optimal solutions. The comparation was performed with CPU time stopping condition: all the heuristics were allowed to run until 20 restarts of Mltistart Local Search (MLS). It appeared that VNS performs best in average on all types of random task graphs. Since we generated random task graphs with known optimal solutions, we could discuss the quality of obtained heuristic solutions for these graphs. The disappointing fact was that for some of the test instances the best solutions (usualy produced by the use of VNS) were more than 50% worse than the optimal one. In order to explaine this deviation, we

* This research has been supported in part by NSF Serbia. The first author has been supported by grant no. 04M02C. The third one by grant no. 04M03C.

performed experimental analysis of MSPCD problem varying the values of VNS parameters.

The paper is organized as follows. In the next section we describe the implementation of Local Search (LS) to MSPCD, solution representation, neighborhood structures and search parameters. The test instances that turn out to be hard for scheduling are descused in section 3. The section 4 contains analysis of the influence of communication delays to solving MSPCD, the obtained topology of local minimas, the efect of selected initial solution and VNS parameters. Section 5 concludes this paper.

2. IMPLEMENTATION DETAILS

In our implementation of heuristics based on LS procedure (TS, VNS, MLS) we defined solution space to be the set of all permutation of tasks. Feasible solutions are feasible permutations, i.e. tasks orderings that obey precedence relations between tasks defined by the task graph. Starting from a permutation, the actual solution (schedule, i.e. the index of processor and the starting time instance for each task) can be determined by the use of some scheduling rule. We choose Earliest Start (ES) rule since it has been widely used in the literature. The idea of ES rule is that a task has to start execution as soon as posible, because then it will be finished earlier and its succeeding tasks can begun their execution earlier.

After the ES rule is applied, we can calculate the objective function value, i.e. schedule length. Obviously, we have implicit connection between solution representation and the value of the objective function. The advantage of representing solution as feasible permutation is efficient neighborhood searh, since there are a lot of neighborhood structures that can be applied. The disadvantage of such representation is that several different permutations may produce the same schedule (after applying ES). The ilustration of these observation is given on Fig. 1. We performed scheduling procedure for all 1-Swap neighbors of known optimal solution. As a result, we obtained 679 optimal schedules versus 437 nonoptimal ones. Then, we sorted the obtained nonoptimal schedules in nondecreasing order and reperesent them graphicaly on Fig. 1. As can be seen on Fig. 1 the different permutations produce the same nonoptimal schedule; also worst solution in 1-Swap neighborhood has 90% error.

In LS procedures we have to obey precedence constraints between tasks in order to assure correct execution of paralellized task graph. We made experiments with several neighborhood structures always using the reduced versions. For example, reduced 1-Swap neighborhood search is as follows: for each task in current solution we perform all feasible changes of its position in this solution in forward (or backward direction). This means that task can not move to the right from any of its successors (or to the left from its predecesors). The search direction can be seen as a parameter. We also experimented with best and first improvement search strategies and conclude that first improvement is faster and gives us better results in average.

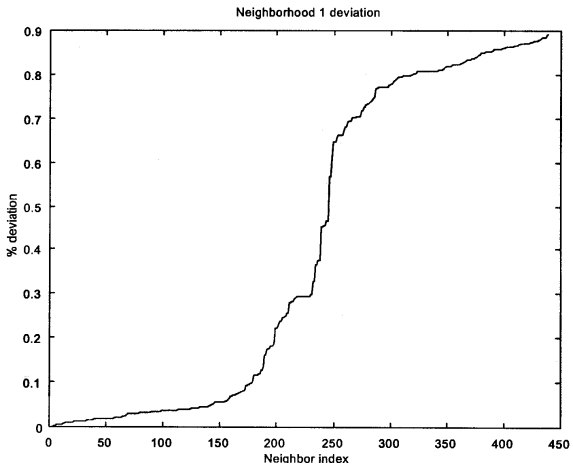

Fig. 1 % deviation from the optimal value in the 1-Swap neighborhood of the optimal solution

Except 1-Swap neighborhood structure, we used restricted versions of Or-opt, which is an ordered combination of 3-Swap, 2-Swap and 1-Swap. The 3-Swap neighborhood means that we take 3 succeeding tasks and move them to all possible (feasible) positions trying to minimize scheduling length. Similarily, the 2-Swap neighborhood is obtained by all possible changings of positions of 2 succeeding tasks. The order of neighborhoods in Or-opt is defined by the change significance and neighborhood sizes. This means that we try to improve our objective function value by coarse moves first, and then performe the fine moves. At the same time, we search in the small neighborhoods first.

The last neighborhood we implemented in [2] is interchange, it is obtained by all possible (feasible) exchanges of positions of any two tasks which is the special case of 4-Opt move.

The shaking procedure of the VNS [4,6] is realized in two variants: 1) *k*-Swap, meaning that *k* times we pick up a task and change its position in feasible permutation and 2) *k*-interchange, which consists in *k* interchanges performed on current incumbent solution. In our

pervious paper [2] we made experiments only with single neighborhoods, single directions of search procedures and unique stopping criterion (CPU time). We selected the best obtained parameter combination for our examples and made a conclusion that among VNS, TS, MLS and even GA, VNS performes best in average. Inspite this, the obtained schedules for tasks graphs (generated as explained in next Section) were more than 50% worse than the known optimal solutions.

3. DESCRIPTION OF 'HARD' TEST INSTANCES

The task graphs with known optimal solutions were generated in such a way that a load balance between processors is achieved and no idle time intervals occure in the optimal solution. The communication is suppossed to be intensive between tasks executed on the same processor. The generation algorithm is described in [5].

Dense task graphs (Ɉ=0.67) are not difficult for scheduling since the data dependences force the scheduling proces to minimize communication delays. So, in these examples optimal solution is obtained as the initial one (permutation defined by the CP method is scheduled according to ES rule) or, at the worst, when the 1-Swap LS is applied to the initial solution. Sparse task graphs (Ɉ=0.33) are characterized by a large number of feasible permutations and the initial one (obtained by the CP method) gives the solution at relatively poore quality (since there are a lot of independent tasks that can be scheduled in an arbitrary way). Acording to a large number of feasible permutations (representing the solution space) it is impossible to search the solution space efficiently.

4. EXPERIMENTAL ANALYSIS

In this section we investigate the influence of the task graph structure and search parameters to the efficiency of our scheduling procedures. Experiments are performed on task graph containing n=200 tasks with known optimal schedule length equal to 1200 computing cycles obtained by scheduling feasible permutation 1,2,é ,200 using the ES rule.

We restrict our attention to sparse graphs (Ɉ=0.33). The best obtained schedule was 1501 computational cycles which is 25% worse than the optimal one. This schedule was obtained by the use of first improvement Or-opt LS performed in forward direction, within CPU=10min and k_{max}=10. We try to explain this deviation and to improve the scheduling results by combining neighborhoods and search directions. Let us first determine the influence of communication delays to scheduling process.

4.1. Effect of communication delays

To investigate the influence of communication delays to quality of heuristic schedule we performed the same procedure as for composing Fig. 1. for the case with neglected communication time.

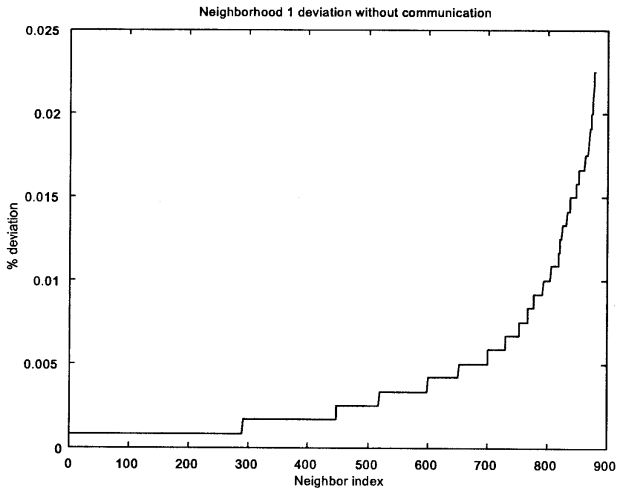

Fig. 2. The schedule deviation without communications

We obtained only 237 optimal schedules and 879 nonoptimal ones with deviations of nonoptimal solutions from the optimal one represented on Fig. 2. First, we conclude that in the presence of communication delays more optimal neighbors are found since the tasks were forced to be scheduled in such a way as to avoid expensive communications. Next, it can be seen that the deviation of the nonoptimal schedules from the optimal ones is much larger for the case with communication delays. If communication is negligible, the tasks can be packed with no delays.

4.2. Effects of neighborhood structures

In our VNS approach [2] we have use several neighborhood structures: 1-Swap, Interchange and Or-opt. Our first experiment consider one neighborhood structure at a time and all results differ one from another 5% at most. Then we combined the neighborhoods, i.e. we apply Variable Neighborhood Descent (VND) approach within VNS. In that way we achieve significant improvements. In Table 1 are given results for different combination of neighborhood structures. The last column contains the scheduling length obtained when the neighborhoods are searched in both directions (forward-backward), by the first improvement rule. In all these examples we fixed k_{max} to 10 and t_{max}=10min.

Table 1: Objective values for n=200 in t_{max}=10min

	1	2	3	4
	1-Swap	Or-opt	Int.ch+2	FB+3
f	2009	1501	2131	2120

As can be seen from this table in first 10min the Or-opt is the most efficient neighborhood structure.

4.3. Effects of t_{max}

To compare different heuristics in solving scheduling problem, we choose given CPU time (t_{max}) to be the stopping criterion. Since poor solutions were obtained within t_{max}=10min, we decided to investigate the influence of t_{max} as well. The experimental results are shown in Table 2. Here, k_{max}=20.

Table 2: The influence of t_{max} for n=200

t	10min	1h	2h	3h
2	1501	1435	1435	1435
4	2120	1416	1317	1279

From the results given in this table we can conclude that after initial improvements, Or-opt search stucks in poor local minima. On the contrary the sequence of several neighborhoods (VND as LS within VNS) allows us to obtain the solution which is about 6% worse than the optimal one within CPU time of 2.5 hours.

4.4. The influence of k_{max}

Table 3: The influence of k_{max} for n=200

k_{max}	5	10	15	20
F	1534	1312	1627	1279

The main parameter of VNS metaheuristic is maximal number of neighborhoods used for shaking. In Table 3 we present the experiments with changing k_{max} in best variant of VNS (i.e. VND with Interchange+Or-opt+FB) The t_{max}= 3h.

4.5. Effects of other parameters

The investigation of initial solution, the neighborhood structure for shaking, the k_{step} parameter, the *plato* parameter gave no significant improvements in the scheduling results, and thus we did not include them in the final version.

5. CONCLUSION

In this paper we investigated reasons why VNS, the most succesfull heuristic for solving MSPCD, give results above optimal with average error larger than 50%. We concluded that using better estimation of parameters could reduce deviation to 6%. More specifically, increasing the CPU time allowed in the search (t_{max}) and increasing the number of neighborhood structures in VND (k'_{max}) influence the most performances of VNS. Future work could consist of implementation of more neighborhood structures in order to reduce the gap more.

REFERENCES

[1] I. Ahmad and M. K. Dhodhi. Multiprocessor scheduling in a genetic paradigm. *Parallel Computing*, 22:395-406, 1996.

[2] T. DavidoviĹ, P. Hansen and N. MladenoviĹ. Variable neighborhood search for multiprocessor scheduling problem with communication delays. *IV Int. Metaheuristics Conf. MIC 2001,* July, 16-21. 2001. (accepted).

[3] T. DavidoviĹ and N. MladenoviĹ. Genetic algorithms for multiprocessor scheduling problem with communication delays. *In Proc. X Yug. Math. Congr.*Jan. 21-24. 2001.

[4] P. Hansen and N. MladenoviĹ. Variable neighborhood search; Principles and applications. *Europ. J. Operational Research*, 24(11):1097-1100, 1997.

[5] Y.-K. Kwok and I. Ahmad. Efficient scheduling of arbitrary task graphs to multirocessors using a parallel genetic algorithm. *J. Parallel and Dist. Comp.,* 47:58-77, 1997.

[6] N. MladenoviĹ and P. Hansen. Variable neighborhood search. *Comp.Oper. Res.* 24(11):1097-1100, 1997.

[7] S.C.S. Porto and C.C. Ribeiro. A tabu search approach to task scheduling on heterogeneous processors under precedence constraints. *Int. J. High Speed Computing,* 7:45-71, 1995.

[8] S.C.S. Porto and C.C. Ribeiro. Parallel tabu search message passing synchronous strategies for task scheduling under precedence constraints. *J. Heuristics,* 1:207-223, 1995.

[9] A. Thesen. Design and evaluation of a tabu search algorithm for multiprocessor scheduling. *J. Heuristics*, 4(2):141-160, 1998.