SOLVING UNRESTRICTED PARALLEL MACHINE SCHEDULING PROBLEMS VIA EVOLUTIONARY ALGORITHMS

Gatica C., Ferretti E., Gallard R.
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)[1]
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106
(5700) - San Luis -Argentina
e-mail: ´{crgatica, ferretti, rgallard@unsl.edu.ar}
Phone: +54 2652 420823
Fax : +54 2652 430224

Abstract

Parallel machine scheduling, also known as parallel task scheduling, involves the assignment of multiple tasks onto the system architecture's processing components (a bank of machines in parallel).

A basic model involving *m* machines and *n* independent jobs is the foundation of more complex models. Here, the jobs are allocated according to resource availability following some allocation rule. The completion time of the last job to leave the system, known as the makespan (C_{max}), is one of the most important objective functions to be minimized, because it usually implies high utilization of resources, but other important objectives must be also considered. These problems are known in the literature [9, 11] as *unrestricted parallel machine scheduling problems*. Many of these problems are NP-hard for $2\leq m \leq n$, and conventional heuristics and evolutionary algorithms (EAs) have been developed to provide acceptable schedules as solutions.

This presentation shows the problem of allocating a number of non-identical independent tasks in a production system. The model assumes that the system consists of a number of identical machines and only one task may execute on a machine at a time. All schedules and tasks are non-preemptive. A set of well-known conventional heuristics will be contrasted with evolutionary approaches using multiple recombination and indirect representations.

[1] The LIDIC is supported by the Universidad Nacional de San Luis and the ANPCYT (National Agency to Promote Science and Technology).

1. Introduction

Unrestricted parallel machine scheduling problems are common problems in production systems. In the literature, minimization of the makespan is extensively approached and benchmarks can be easily found. This is not the case for other important objectives such as the due-date-related objectives (e.g. maximum, average and weighted values for lateness, tardiness and number of tardy jobs). Various heuristics, which fits differently on different problems, have been developed. Some of them are: longest processing time (LPT), weighted longest processing time (WLPT), shortest processing time (SPT), weighted shortest processing time (WSPT), earliest due date (EDD), least slack (Slack), Hodgson´s algorithm (HDG), weighted Hodgson (WTD HDG), Rachamadugu and Morton (R&M).

In previous research we have been working with evolutionary approaches to solve the Pm| |C_{max} problem and contrast their results against LPT for makespan minimization [6]. At this time we want to extend our work to Pm| |Obj, where *Obj* is some of the above indicated due-date-related objectives and contrast the EAs against those conventional heuristic which would perform better.

Because well-known experimental repositories do not provide testing cases, to extend our work we need to develop our own benchmarks. The idea is to collect data, which have been designed for distinct scheduling due date related problems and use it in our algorithms. In this way due dates, processing times and arrivals are proved to be consistent.

The next step is to run a software package, *Parsifal*, provided by Morton and Pentico [11], to determine the best-performers among the conventional heuristics and to contrast them against our EAs.

2. The evolutionary multirecombined approach

The evolutionary algorithms to be developed belong to the multirecombined family. We describe now their main characteristics. Multiple Crossovers per Couple (MCPC) [7] and Multiple Crossovers on Multiple Parents (MCMP) [8] are multirecombination methods, which improve EAs performance by reinforcing and balancing exploration and exploitation in the search process. In particular, MCMP is an extension of MCPC where the multiparent approach of Eiben [3, 4, 5] is introduced. Results obtained in diverse single and multiobjective optimization problems indicated that the searching space is efficiently exploited by the multiple application of crossovers and efficiently explored by the greater number of samples provided by the multiple parents.

A further extension of MCMP is known as MCMP-SRI (stud and random immigrants) [10]. This approach considers the mating of an evolved individual (the stud) with random immigrants. The process for creating offspring is performed as follows. From the old population the stud is selected and inserted in the mating pool. The number of parents in the mating pool is completed with randomly created individuals (random immigrants). The stud mates every other parent a number of times. The best offspring is inserted in the new population.

A latest extension MCMP-SRSI (stud, random and seed immigrants) includes problem-specific-knowledge in the EA algorithm, seeds are recombined in the evolutionary process. Seeds are good solutions provided by heuristics specifically designed for the problem under our concern and they can help the evolutionary process by guiding the search more rapidly towards the promising areas of the solutions space.

3. Representations

From the representation perspective many evolutionary computation approaches to the general scheduling problem exists. With respect to solution representation these methods can be roughly categorized as *indirect* and *direct* representations (Bagchi et al, 1991 [1], Bruns R. 1993 [2]).

In the case of indirect representation of solutions the algorithm works on a population of encoded solutions. Because the representation does not directly provides a schedule, a schedule builder is necessary to transform a chromosome into a schedule, validate and evaluate it. The schedule builder guarantees the feasibility of a solution and its work depends on the amount of information included in the representation. We decided to use three approaches with *indirect* representation; the first belongs to the *indirect-permutation* representation class, and the other two are of the *indirect-decode* representation class.

Permutation-based representation

In this approach we use permutations, which describe a list of tasks priorities. The schedule builder takes the first ready task from the list, and assigns it to an available processor. If two or more processors are available, the one with lowest identifier processes the ready task.

Decoder-based representation

proc →	1	2	3	2	1	3	1	2
task →	1	2	3	4	5	6	7	8

Fig. 1. Chromosome structure for the task allocation problem

In the first decoder-based approach, called *processor dispatching priorities indirect-decode representation*, a schedule is encoded in the chromosome in a way such that the task indicated by the gene position is assigned to the processor indicated by the corresponding allele, as indicated in Fig.1. Here the chromosome gives instructions to a decoder on how to build a feasible schedule. In our case the decoder is instructed in the following way: By following the task priority list, traverse the chromosome and assign the corresponding task to the indicated processor as soon as it is available. In this way a priority for processors is established telling which processor is assigned to which job. Even if other processors are available the job waits until the designed processor becomes available. Regarding tasks, a task priority list is defined at the beginning and remains the same during the search (i.e. canonical order).The processor priorities for dispatching tasks (chromosomes) change while searching for an optimal schedule (minimum makespan).

In the second decoder-based approach, called *task priority list indirect-decode representation* each individual in the population represents a list of task priorities. Each list is a permutation of task identifiers. Here a chromosome is an n-vector where the i^{th} component is an integer in the range 1..(n – i +1). The chromosome is interpreted as a strategy to extract items from an ordered list L and builds a permutation. We briefly explain how the decoder works. Given a set of tasks represented by a list L and a chromosome C, the gene values in the chromosome indicate the task positions in the list L. The decoder builds a priority list L′ as follows: traversing the chromosome from left to right it takes from L those elements whose position is indicated by the gene value, puts this element in L′ and deletes it from L (shifting elements to left and reducing L length). It continues in this way until no element remains in L. Fig. 2 shows the list L, the chromosome C and the resulting priority list L´.

L =	1	2	3	4	5	6
chromosome C =	3	2	2	1	1	1
L' =	3	2	4	1	5	6

Fig.2. L is the task list, L' is the priority task list.

4. Genetic operators

The kind of representation for a chromosome we choose in an evolutionary algorithm imposes constraints on the genetic operator we will use and this fact influence the search process.

To obtain valid offspring and avoid repair actions, especial crossover operators such as PMX, OX or CX, must be used in *permutation-based* representation. This will ensure that a child will also be a permutation. PMX crossover and exchange mutation, are natural and efficient proved operators. This approach is also natural for MCPC, where once selected a pair of parents, multiple crossover are applied by defining different cut points. But in the case of MCMP, the exchange of genetic material is limited to pairs of parents from the mating pool. This means that to create offspring all possible pairs could be formed, obtaining 2 children from each pair after a single crossover operation. In both cases (MCPC and MCMP), after crossover, one or more children are selected according with some criteria, to insert in the next population. The process is repeated as many times as needed to complete the population size.

Decoder-based representation is more flexible. Most common operators produce valid offspring. Children from decoders are decoders. In the case of MCPC, any of the traditional crossovers (one-point, multipoint, uniform, etc) can be safely used, while for mutation little creep, big creep and exchange are of common use. In the case of MCMP, two of the scanning crossover (SX) family; *uniform scanning crossover* (USX), and *fitness based scanning* (FBSX) could be used. In this case for the exchange of genetic material all parents from the mating pool contribute as donnors, simultaneously in a single crossover operation creating a single child. Again in both cases (MCPC and MCMP), after crossover, selection of children and completion of the next population is done as described above.

5. Current and future work

At this time the first versions of the algorithms for a variety of objective functions and instances of different problem sizes are being implemented and tested for different parameter settings in the static scheduling domain.
In the future larger problem sizes will be studied and partially or totally dynamic scheduling problems will be faced to contrast evolutionary algorithms against conventional approaches.

6. References

[1] Bagchi S., Uckum S., Miyabe Y., Kawamura K. – *Exploring problem-specific recombination operators for job shop scheduling*- Proceedings of the Fourth International Conference on Genetic Algorithms, pp 10-17, 1991.

[2] Bruns R. – *Direct chomosome representation and advanced genetic operators for production scheduling*. Proceedings of the Fifth International Conference on Genetic Algorithms, pp 352-359, 1993.

[3] Eiben A.E., Raué P.E., and Ruttkay Z., "Genetic algorithms with multi-parent recombination", *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, Springer-Verlag, 1994, number 866 in LNCS, pp. 78-87.

[4] Eiben A.E., Van Kemenade C.H.M., and Kok J.N., "Orgy in the computer: Multi-parent reproduction in genetic algorithms". Proceedings of the *3rd European Conference on Artificial Life*, Springer-Verlag, 1995, number 929 in LNAI, pages 934-945.

[5] Eiben A.E. and. Bäck Th., "An empirical investigation of multi-parent recombination operators in evolution strategies". Evolutionary Computation, 5(3):347-365, 1997.

[6] Esquivel S., Gatica C., Gallard R. – "Evolutionary Approaches with Multirecombination for the Paralell Machine Scheduling Problem", XX Conferencia Internacional de la Sociedad Chilena de Ciencias de la Computación, Noviembre 2000, Santiago, Chile. IEEE Publishing Co, págs. 1 – 6.

[7] S. Esquivel, A. Leiva, R. Gallard, "Multiple Crossover per Couple in Genetic Algorithms", Proceedings of the *Fourth IEEE Conference on Evolutionary Computation (ICEC'97)*, Indianapolis, USA, April 1997, pp 103-106.

[8] S. Esquivel, H. Leiva,.R. Gallard, "Multiple crossovers between multiple parents to improve search in evolutionary algorithms", Proceedings of the *Congress on Evolutionary Computation (IEEE)*. Washington DC, 1999, pp 1589-1594.

[9] T. Morton, D. Pentico*, "Heuristic scheduling systems"*, Wiley series in Engineering and technology management. John Wiley and Sons, INC, 1993.

[10] Pandolfi D., De San Pedro M., Villagra A., Vilanova G., Gallard R.- " Studs mating immigrants in evolutionary algorithm to solve the earliness-tardiness scheduling problem" . Aceptado para publicación en Cybernetics and Systems del Taylor and Francis Journal, (U.K.) July 2002.

[11] Pinedo M, *Scheduling Theory, Algoritms, and Systems.* Prentice-Hall -1995. mpinedo@stern.nyu.edu