European Journal of Operational Research 268 (2018) 463–472

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

The Unit-capacity Constrained Permutation Problem

Pascale Bendotti[a,b], Pierre Fouilhoux[b,*], Safia Kedad-Sidhoum[b]

[a] EDF R&D, Département OSIRIS, 1 avenue du Général de Gaulle, Clamart cedex 92141, France
[b] Sorbonne Universités, Université Pierre et Marie Curie, LIP6 CNRS UMR 7606, 4 place Jussieu, Paris 75005 France

ARTICLE INFO

Article history:
Received 23 July 2016
Accepted 27 January 2018
Available online 3 February 2018

Keywords:
Combinatorial optimization
OR in energy
Complexity theory
Dynamic programming
Dominance and symmetry properties

ABSTRACT

The Unit-capacity Constrained Permutation Problem (UCPP) is to find a sequence of moves for pieces over a set of locations. From a given location, a piece can be moved towards a location with a unit-capacity constraint, i.e. the latter location must be free of its original piece. Each piece has a specific type and at the end every location must contain a piece of a required type. A piece must be handled using a specific tool incurring a setup cost whenever a tool changeover is required. The aim of the UCPP is finding a sequence of moves with a minimum total setup cost. This problem arises in the Nuclear power plant Fuel Renewal Problem (NFRP) where locations correspond to fuel assemblies and pieces to fuel assembly inserts. We first show that the UCPP is NP-hard. We exhibit some symmetry and dominance properties and propose a dynamic programming algorithm. Using this algorithm, we prove that the UCPP is polynomial when two tools and two types are considered. Experimental results showing the efficiency of the algorithm for some instances coming from the NFRP are presented.

1. Introduction

We consider a set of p *pieces* that have to be moved over a set of locations. A location contains at most one piece. Each piece possesses a specific *type* among q different types, denoted by $\{1, \ldots, q\}$. In *the initial assignment*, each piece is assigned to an initial location and, in *the final assignment*, each location must contain a piece of a given type. Therefore, for each location whenever the type of a piece in the initial assignment is different from that required in the final assignment, the piece should change location. Finding a sequence of moves becomes an issue when pieces are handled with a unit-capacity constraint. In fact each move involves only one piece at a time, and assigns a piece to another location provided the piece originally there has previously been moved. It is assumed that each piece can be moved only once and that any location can be reached from any other location. Furthermore, to handle a piece of type $t \in \{1, \ldots, q\}$, a specific tool is used among the r possible tools among $\{1, \ldots, r\}$. Therefore, pieces that can be handled in turn by using the same tool are gathered into batches, and each batch of pieces will incur a *setup cost*. The *Unit-capacity Constrained Permutation Problem* (UCPP) is to find a sequence of moves with a unit-capacity constraint for pieces to have the required type, such that the total setup cost is minimized.

The time it takes to pick-up, move and drop-off a piece is assumed to be constant. We further assume that the setup cost is the same for every tool and is equal to one. It follows that the setup cost is equal to the number of tool changeovers. Moreover we assume that each instance of the problem supports at least one feasible solution.

To illustrate consider an instance with eight locations and five pieces. Fig. 1(a) shows the location (represented by a square) of each piece in the initial assignment. The five pieces are of three different types a, b and c (symbolized by circles). The pieces of type a can be moved using tool 1 (circles with full line), while the pieces of types b and c can be moved using tool 2 (circles with dashed line). Fig. 1(b) shows the type of piece required for each location in the final assignment. One of the two pieces of type c can be moved to locations 7 or 8. However none of the two pieces of type a can be moved to location 2 or 6, as the original pieces are still there.

Therefore a particular sequence of moves is necessary to obtain the final assignment from the initial one. A possible sequence involving five moves (i.e. one move per piece) can be described as follows. The first move is with the piece of type b using tool 2 from 2 to 3, thus making it possible to move a piece of type a using tool 1 from 1 to 2. The following two moves are from 6 to 7 and from 5 to 6. The final move from 4 to 8 leads to the final assignment shown in Fig. 1(b). Such a sequence of moves requires

* Corresponding author at: Université Pierre et Marie Curie, Laboratoire LIP6, Boîte courrier 169, 4 Place Jussieu, Paris cedex 05, 75252, France.

E-mail addresses: pascale.bendotti@edf.fr (P. Bendotti), pierre.fouilhoux@lip6.fr (P. Fouilhoux), safia.kedad-sidhoum@lip6.fr (S. Kedad-Sidhoum).

https://doi.org/10.1016/j.ejor.2018.01.049
0377-2217/© 2018 Elsevier B.V. All rights reserved.

five tool changeovers. Note that there exists a sequence involving less tool changeovers.

The UCPP arises in the nuclear power plant fuel renewal context. The fuel assemblies are located in the core of the reactor. Depending on the length of an operating cycle that ranges from 12 to 18 months, a fraction typically one third or one quarter of the fuel assemblies is replaced in the core. During a refueling outage, some spent fuel assemblies are then replaced with fresh ones while the remaining fuel assemblies have been used from one cycle up to two or three. Each fuel assembly is equipped with an insert. In particular, specific requirements apply depending on the location of the inserts in the core and translate into constraints on types of inserts. The *Nuclear power plant Fuel Renewal Problem* (NFRP) aims at performing the permutation of the inserts over the spent, fresh and remaining fuel assemblies. As an insert consists of a cluster of long rods, it should be handled with both great precision and care. For these reasons, only one insert can be moved at a time by a spent fuel mast bridge, i.e. the fuel handling system. A specific tool is required to handle a subset of types of inserts. Moreover the bridge carries only one tool at a time. This makes the constraint on the capacity quite restrictive as not only one insert can be moved at a time, but also the targeted location should be free of insert, in particular if it contains one. In the whole refueling operation, the refueling machine lifts every fuel assembly out of the reactor core and moves it from the reactor building through a transfer container to the fuel building. Installing new fuel assemblies in the core is performed using the same process in reverse. From a practical point of view, the NFRP is solved before the fuel assemblies are moved out of the reactor core. The fuel assemblies will be located in the fuel building according to the moves involved in the solution of the NFRP. This makes the assumption on a constant time to move an insert reasonable. Consequently, the NFRP aims at minimizing the number of tool changeovers which is of paramount importance in the nuclear power plant fuel renewal context.

The NFRP reduces to the UCPP where the locations correspond to the fuel assemblies, and the pieces to the fuel assembly inserts and is to find an optimal handling sequence for the spent fuel mast bridge to complete the permutation of inserts for the renewal of the fuel assemblies.

For the nuclear power plants operated by Electricité de France, a heuristic developed at EDF R&D in the seventies is used (Guigou & Mauget, 1987; Mauget, 1987). It is nowadays used for the refueling of the 58 nuclear reactors operated by EDF. The heuristic assumes only one final location per insert. This restriction may lead to a sub-optimal solution. The assumption is already too restrictive for the operating reactors. It is also the case for the coming units of the European Pressurized Reactors (EPR) as new types of pieces with several final locations must be considered. This involves UCPP instances containing much more symmetries. In terms of practical setting, there is a need for more accurate and efficient solving approaches to tackle these new NFRP instances.

The motivation of this article is twofold. First it is to rely on a complexity analysis of the UCPP problem in deciding whether some exact polynomial methods can be derived. Second it is to exhibit symmetries appearing within new NFRP instances and to derive a dedicated dynamic programming approach taking benefits from symmetries as well as dominance properties.

The organization of the article is as follows. Section 2 proposes a literature review related to the UCPP together with a description of the contributions of this article. Section 3 provides some notations and properties used in Section 4 to set the complexity of the UCPP. Some particular cases are exhibited. In Section 5, some dominance properties are introduced to reduce the solution space. Section 6 shows that the UCPP instances feature some symmetries. Section 7 presents a dynamic programming algorithm exploiting symmetries and dominance properties. Using this algorithm, a particular case of the UCPP is proved to be polynomial. Finally Section 8 presents some experimental results while Section 9 concludes the article.

2. Literature review and contributions

The UCPP tends to remind us of combinatorial problems dealing with robots in production scheduling. However the unit-capacity constraint involved in the UCPP translates into the use of a single robot to move all the pieces. Furthermore unlike many production scheduling problems where the robot carries a subset of tools (see Crama, 1997), the unit-capacity constraint implies that the robot can carry only one tool at a time.

The UCPP can be seen as a particular *Pickup and Delivery problem* (PDP) (Savelsbergh & Sol, 1995). In the PDP, a vehicle must satisfy a set of transportation requests where a request is defined by a pick-up point, a corresponding delivery point and a demand to be carried between these two locations. Hence the UCPP seems to be closely related to the PDP using a single vehicle with a unit-capacity constraint (Gribkovskaia, Laporte, & Shyshou, 2008) and, more specifically, to the *swapping problem* (Anily & Hassin, 1992). The swapping problem can be seen as a particular PDP where every pick-up point of a request is also a delivery point of another request. In the preemptive (non-preemptive) swapping problem, a piece can (cannot) be temporarily stored at intermediate vertices on the vehicle's route before reaching its final destination and can (cannot) be reloaded later (Anily, Gendreau, & Laporte, 2011; Bordenave, Gendreau, & Laporte, 2009; Bordenave, Gendreau, & Laporte, 2012; Erdoğan, Cordeau, & Laporte, 2010). The UCPP is then related to the non-preemptive version of the swapping problem. However the UCPP deeply differs from these two former problems. For a pick-up-and-delivery operation, the vehicle would frequently have to handle two pieces at a time. In the swapping problem, it is allowed to swap pieces between two locations. Neither the former nor the latter operation is possible in the UCPP. Another difference lies in the objective function: while in both the PDP and the swapping problem the cost matrix corresponds to the distance between locations, and is assumed to satisfy the triangular inequalities, in the UCPP the total setup cost is minimized, and the total distance traveled not considered.

As only tool changeover (setup) cost minimization is considered, the UCPP falls into the single-machine scheduling problem with sequence-dependent setup cost category (Allahverdi, Ng, Cheng, & Kovalyov, 2008), which can be reformulated as a Traveling Salesman Problem (TSP). It is worth noting that Van der Veen and Zhang (1996) considered a special case of a sequence-dependent scheduling problem, referred as the K-group case, where jobs can be divided into K groups depending on which resource they require. The authors proposed a polynomial-time algorithm to solve this problem. Furthermore, Psaraftis (1980) used a dynamic programming approach for sequencing a given set of jobs on a single machine for the K-group case. For a problem of m jobs per group, the algorithm's running time is shown to grow as $K^2(m+1)^K$. This is a polynomial function of m, but exponential function of K if K is not fixed. Such an algorithm could be of practical interest for applications where K is small. The UCPP can be seen as a single-machine scheduling problem where the pieces are the jobs to process and the required resource the tool. Recall that in the UCPP, a piece can only be assigned to a given location if the piece originally there has already been moved. Therefore, precedence constraints should be considered and the algorithms proposed in Van der Veen and Zhang (1996) are not relevant for the UCPP.

In Bendotti and Fouilhoux (2016), the authors present a necessary and sufficient condition to check whether an instance of the UCPP is feasible or not. In this latter case, they propose to use