Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 05-2023
DOI 10.17605/OSF.IO/KN46A

AN ITERATIVE THREE-STAGE NEIGHBORHOOD SEARCH FOR SOLVING PRECEDENCE CONSTRAINED AGRICULTURAL LAND INVESTMENT PROBLEM

RAFID SALIH SARHAN
Department of Electrical and Computer Engineering, University of Duhok, Zakho Street, Duhok, Kurdistan region, Iraq. Email: Rafid@uod.ac

SAGVAN ALI SALEH
Department of Electrical and Computer Engineering, University of Duhok, Zakho Street, Duhok, Kurdistan region, Iraq. Email: sagvan.saleh@uod.ac

Abstract

The use of neighborhood search techniques to address a practical issue faced by agricultural investors is examined in this study. The problem is named as agricultural land investment problem with precedence constraints and it has an important impact on the agriculture issues. The tackled problem can be viewed as a variant of the well-known classical 0-1 knapsack problem where precedence constraints are imposed on pairs of items. Precedence constraints taking into account a precedence relation between items. This paper first simulates the considered problem as precedence constraints knapsack problem and presents a mathematical representation model to represent it. Then, an iterative three-stage neighborhood search method is proposed for optimizing the problem. The proposed method consists of three stages. First stage applies a greedy procedure in order to construct a feasible solution. Second stage applies local search procedures in order to enhance the quality of the solutions at hand. Third and last, in order to broaden the search space, a random neighborhood destruction approach is introduced. Finally, the effectiveness of the suggested approach is assessed and contrasted with the outcomes obtained by greedy and local search techniques. The presented method is competitive and efficient since it produces excellent solutions in a reasonable amount of time.

Keywords: Precedence Constraints, 0-1 Knapsack, Heuristic, and Neighborhood Search

1. INTRODUCTION

Numerous real-world scenarios can be represented as combinatorial optimization problems in order to be solved. One of these situations belongs to agriculture, where a large agricultural land need to be invested. The problem is that, there are numerous plant varieties that can be grown with little money and time. The price of raising each plant varies. However, every one of them is making profit. In addition, according to the needed of some farmers, precedence constraints are imposed, i.e., there are several chains of plants must be considered (Nancel-Penard, et al., 2022). However, the objective is to maximize the profit of land investment with the considering of precedence constraints (Aslan, et al., 2023). This problem is a variant of agricultural land investment problem (ALIP) presented by Saleh in (Saleh, 2018) with the variant, where precedence constraints have been presented and imposed on pairs of items. Precednce constrainted taking into account a precedence relation between items, i.e., some items must precede some others (Samphaiboon & Yamada, 2000). The tackled problem is named as the precedence constraints agricultural land investment problem (abbreviated to PCALIP). This paper investigates the use of neighborhood search techniques for

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 05-2023
DOI 10.17605/OSF.IO/KN46A

optimizing the considered problem. As is obvious, the PCALIP is an NP-hard problem. The precedence constraints knapsack problem (abbreviated PCKP) is a well-known combinatorial optimization problem that can be used to simulate the PCALIP in order to streamline the treatment of the issue. In fact, there are numerous real-world scenarios that can be recreated as members of the KP family in a range of fields, including the computer sciences. (Kellerer, et al., 2014).

The PCALIP is characterized by a knapsack of capacity c; a set I of n items, and a set E of precedence relationships imposed on items, where $E \subseteq I \times I$. A precedence relationship $(i, j) \in E$ exists if item i can be selected and placed in the knapsack only if item j is in the knapsack. Each item $i \in I$ is represented by a nonnegative weight w_i and a profit p_i. The PCALIP is the problem of maximizing the total profit of products that can fit in the knapsack and whose combined weight does not go above the knapsack's carrying capacity while also satisfying the order of precedence. In order to tackle and optimize the considered problem, it is simulated as an optimization problem, known as the precedence constrained knapsack problem (Boland, et al., 2012). Therefore, the mathematical representation of PCALIP can be written as follows:

$$Max: \quad f(x) = \sum_{i=1}^{n} p_i \, x_i \quad \text{.......... (1)}$$

$$s.t. \quad \sum_{i=1}^{n} w_i \, x_i \leq c \quad \text{........... (2)}$$

$$x_i \leq x_j \quad \forall (i, j) \in E \quad \text{........... (3)}$$

$$x_i \in \{0,1\} \quad \forall i \in I = \{1, \dots, n\} \quad \text{........... (4)}$$

Where $x_i, \forall i \in I$, is equal to 1 if the i-th item is placed in the knapsack (i.e., selected in the solution); and 0 otherwise. Equation (1) is the objective function, where the goal is to maximize the total profit. Equation (2) represent the knapsack constraint with the capacity c, which impose that the total weight must not exceed the knapsack capacity. Equation (3) represents the precedence constraints which ensure the precedence relationships. Equation (4) represents the integral constraints (i.e., the item is selected when "$x_i = 1$" or excluded from the solution when "$x_i = 0$"). In order to avoid trivial cases, it is assumed that: the input data $c, w_i, p_i, \forall i \in I$, are positive integers, and $\sum_{i \in I} w_i > c$ (Hifi et al., 2015).

We can observe, from the mathematical representation of the PCALIP, that the solution domain of a 0-1 knapsack problem can be characterized by Equation (2) and Equation (4). By adding, Equation (3) the problem is changed and become another variant of the classical 0-1 knapsack problem, known as the precedence constrained knapsack problem. In other words, the PCALIP reduces to the classical knapsack problem when the precedence constraint is omitted, i.e., $E = \emptyset$ (Boland et al., 2012).

This paper is organized as follows. Section two reviews some related works. Section three introduces a random neighborhood search approach for optimizing the PCALIP. Section four evaluates the performance of the proposed method and analyzes the obtained results. Finally, section five summarizes the contents of the paper.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 05-2023
DOI 10.17605/OSF.IO/KN46A

2. RELATED WORKS

PCALIP is an NP-hard combinatorial optimization problem. It is simulated (in this paper), as the Precedence Constrained Knapsack Problem (abbreviated to PCKP), as illustrated in the previous section. Therefore, the solution procedures dedicated for solving PCKP are also suitable for optimizing the PCALIP (Kellerer et al., 2013).

The literature does, however, include solution strategies that are either accurate or approximatively. Boland et al., in (Boland, et al., 2012) presented an exact method based on clique inequalities for determinng facets of the PCKP polyhedron. Significant reductions in branch-and-bound nodes and overall solution time were achieved by adding these inequalities at the root node. Samphaiboon and Yamada in (Samphaiboon & Yamada, 2000) present exact and approximate methods. They present dynamic programming algorithms as well as preprocessing method to solve PCKP. The dynamic programming can solve small PCKPs instances to optimiality, while by using the preprocessing method they were able to solve the problem with up to 2000 items in few minutes. You and Yamada in (You & Yamada, 2007) present a pegging approach based on Lagrangian relaxation followed by a pegging test. This approach is an exact method where the size of the original problem is reduced to be solved within few minutes. Maiti et al., in (Maiti et al., 2021) presented a specific breakpoint algorithm, which can search appropriate breakpoints of parametric maximum flow related problems. The algorithm is used to solve lagrangian relaxed PCKP as well as linear programming relaxed PCKP in mine pushback design. Then, topo sort is used to produce feasible solutions. In this paper, an iterative three-stages neighborhood search is proposed for optimizing the PCALIP. The first stage serves to construct a feasible solution. The second stage applies local search procedures to enhance the current soltion. Third stage serves to diversify the search space by using a random neighborhood destroying strategy.

3. AN ITERATIVE THREE-STAGE RESOLUTION SEARCH

Neighborhood search methods have already been proven their effecency in developing efficient algorithms to approximate large-scale combinatorial optimization problems (Aarts & Lenstra, 2018). In this paper, a collection of such techniques have been used to optimize the PCALIP. This section presents the main stpes of the proposed solution procedure, which can be viewed as an iterative three-stage neighborhood search. The first stage tries to construct a feasible solution by applying a greedy method. Such a stage construct a solution by solving 0-1 knapsack problem with the precedence constraints. The second stage applies local search in order to improve the solution at hand. Such a stage can be viewed as an intesification stage because it tries to enhance the solutions by applying an exchange technque, where some items are removed to add others. Third stage applies a random destroying strategy by removeing a rather large percentage of items from the current soltuion and degrading it. This stage can be viewd as a diversification stage in order to diversify the search process. The above three stages are iteratived until satisfying a stopping condition.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 05-2023
DOI 10.17605/OSF.IO/KN46A

3.1 First stage to construct a feasible solution

This section demonstrates how to identify a PCALIP solution that is viable. Indeed, the main purpose of the first stage is to construct fast solution, for this reason, among heuristics solution procedures, a greedy procedure is the most suitable choice. Greedy solution procedures can construct a quick fix that is implemented piece by piece and prioritizes an immediate improvement above consequences (Hifi, et al., 2015). In general, this type of neighborhood search technique does not ensure the optimality of the solutions, but it is very fast for determining feasible solutions (Ausiello, et al. 2012). Algorithm 1 illustrates the first stage of the proposed solution procedure for solving PCALIP, where a greedy method is considered. This algorithm is used, mainly, for two aims: (i) to determine a starting solution, and (ii) to complete a destroyed solution obtained from stage 3 (as explained later in section 3.3). In both cases, it yields a fast feasible solution of PCALIP.

The major steps of the proposed greedy strategy are illustrated in Algorithm 1, in which a workable solution is pieced together successively. It start by initialization the problem P_{PCALIP}. Step 3 defines a decision variable x_i, this variable determine whether the $i - item$ is selected or not in the solution. This means that, if $x_i = 1$, the $i - item$ is placed in the knapsack, while if $x_i = 0$, the corresponding item is not selected in the solution, i.e., doesn't placed in the knapsack. Steps 4-11 represents the main loop of the procedure. In this loop, each $i - item$ is selected in a sequentially manner under the following condition: if it is free, i.e., $x_i = 0$. Steps 6-7 ensure that, before putting any item in the knapsack, all its precedence must be selected in the solution, i.e., their $x_i = 1$. Otherwise, stop and try other $i - item$. In all cases, before selecting any item in the solution, the knapsack constraint is checked (see step 5).

Algorithm 1: A feasible solution construction of PCALIP

Input: P_{PCALIP}, an instance of the problem

Output: S_{PCALIP}, a feasible solution

1: Initialize P_{PCALIP},

2: Let i be the total number of items

3: Let x_i be the decision variables of i^{th} items.

4: While $i \geq 0$

5: While ($x_i = 0$ && the knapsack constraint is not violated)

6: Set $x_i = 1$ of all the precedence of $i - item$

7: Set $x_i = 1$ of the $i - item$

8: Update the solution S_{PCALIP}

9: End While

10: $i = i - 1$

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 05-2023
DOI 10.17605/OSF.IO/KN46A

End While

return S_{PCALIP}

3.2 Second stage: local search to find the local optima solution

This section shows how the solution at hand can be enhanced. The main purpose of this stage is to improve the quality of the solution obtained from the first stage. Therefore, a local search method is proposed as an intesification procedure to enhance the solution at hand, and trying to find the local optimum solution (Hifi et al., 2015). The proposed method belongs to neighborhood search methods, which can be viewed as a variant of an exchange technique, where some items are removed from the solution to add others (Aarts & Lenstra, 2018).

Algorithm 2 presents the second stage where a local search procedure is proposed to enhance the current solution.

Algorithm 2: local search method to enhance the solution

Input:S_{PCALIP}, a feasible solution obtained from Algorithm 1

Output: S^*_{PCALIP} an enhanced solution

Let i be the total number of items

let S'_{PCALIP} be a reduced problem

let $\beta = 5\%$ of S_{PCALIP} (i.e., remove 5% of items from the current solution)

While $\beta \geq 0$

Remove $item_{\beta}$ from S_{PCALIP}

Remove all the successors of $item_{\beta}$

Update the reduce problem S'_{PCALIP}

$\beta = \beta - 1$

End While

While $i \geq 0$

If $item_i = 0$ && the knapsack constraint is not violated

Add the precedencies of $item_i$ in S'_{PCALIP}

Add $item_i$ in S'_{PCALIP}

Update S'_{PCALIP}

End If

$i = i + 1$

End While

Return S^*_{PCALIP}

ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 05-2023
DOI 10.17605/OSF.IO/KN46A

Algorithm 2 illustrates the main steps of the proposed local search method. The algorithm start with a solution obtained from Algorithm 1 and tries to enhance it. Steps 4-17 illustrates the main loops in the algorithm. The main idea is that, remove randomly 5% of items from the solution obtained from Algorithm 1 (see steps 4-9), then try to add other items considering the knapsack and precedence constraints (see steps 10-17). This process continues for all not selected items (i.e., $x_i = 0$).

3.3 Third stage: random destroying strategy to diversify the solution space

This section shows how the search process can be diversified in order to escape from a series of local optimum solutions. For this, a random destroying strategy is proposed as a diversification procedure. This strategy tries to explore randomly the sub-solution spaces aiming to find the best local solution (Hifi et al., 2014).

Algorithm 3 illustrates the third stage where a random destroying strategy is proposed to diversify the solution search space.

Algorithm 3: random destroying strategy to diversify the solution search space

Input :S^*_{PCALIP}, an enhanced solution obtained from stage 2

Output: S^d_{PCALIP} a destroyed solution

1: let α be a number of items to be removed from a solution

2: While $\alpha \geq 0$

3: select an $i - item$ randomly from the current solution

4: $x_i = 0$; to remove $i - item$ from the solution

5: Update the destroyed solution S^d_{PCALIP}

6: $\alpha = \alpha - 1$

7: End While

Return S^d_{PCALIP}

Algorithm 3 shows the main steps of the proposed diversification strategy. The algorithm start with the solution obtained from stage 2 and tries to diversify the search process by applying a random destroying procedure. Steps 2-7 shows the main steps in this algorithm. The idea is that, destroy the current solution obtained from stage two by removing $\alpha\%$ of its items (see steps 3-5). The destroyed solution (S^d_{PCALIP}) is then go back to algorithm 1 in order to be completed and provide another feasible solution. This process is iterated until satisfying a stopping condition.

3.4 Overall Algorithm

This section illustrates the overall algorithm proposed in this work: an iterated three-stage neighborhood search for Solving PCALIP.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 05-2023
DOI 10.17605/OSF.IO/KN46A

Algorithm 4 presents the main steps of the overall algorithm. Steps 1-6 shows the main loop, where the three stages are iterated until satisfying the stopping condition. Herein, the stopping condition considered is the number of iterations.

Algorithm 4: an iterative three-stage neighborhood search

Input : P_{PCALIP}, an instance of the problem

Output:S^*_{PCALIP}, the best local solution obtained

1: While stopping criteria is not satisfied

2: Apply Algorithm 1 to determine a feasible solution

3: Apply Algorithm 2 to enhance the solution solution

4: Apply Algorithm 3 to diversify the search process

5: Update S^*_{PCALIP} the best solution found

6: End While

Return S^*_{PCALIP}

4. COMPUTATIONAL RESULTS

This section investigates the effectiveness of the proposed Iterative three-stage Resolution Search (abbreviated to IRSPC) on an instance consists of 1000 items with 55 pairs of precedence relations, which has been generated randomly by using a special program. The algorithm IRSPC was coded in C++ on a computer with Pentium Core i5 CPU at 2.5 GHz.

First step in the computational results investigates the performance of a greedy procedure for solving the PCALIP (Algorithm 1). Recall that, (see Section 3.1), the purpose of this algorithm is to produce a fast feasible solution. This has been achieved as illustrated in Table 1. The algorithm yields an objective value equals to 226362 within 0.015 second.

Table 1: The performance of greedy procedure (Algorithm 1)

Greedy algorithm	
Objective value	226362
Time (s)	0.015 s

Second step in the computational results valuates the effectiveness of the local search procedure, illustrated in Algorithm 2, to enhance the solution at hand. In fact, the proposed algorithm works as an intesification procedure to enhance the solution within short time, by removing 5% of items from it, and add others. This has been achieved as shown in Table 2, where the objective value has been enhanced from 226362 to 271224 within 0.047s.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 05-2023
DOI 10.17605/OSF.IO/KN46A

Table 2: The performance of the local search procedure (Algorithm 2)

	Greedy	**Local search procedure**
Objective value	226362	271224
Time (s)	0.015 s	0.047s

Third step in the computational results investigates the performance of the random destroying strategy, presented in Algorithm 3. In this algorithm, $\alpha\%$ of items are removed randomly from the solution obtained from Algorithm 2, in order to degrade it and escape to other sub solution space. This degradation diversifies the search process and drives the solution procedure to explore randomly a series of solution sub-spaces, trying to escape from a series of local optimum solutions. The destroyed solution is, then, reconstructed again by using Algorithm 1 and 2. The IRSPC is iterated until satisfying the stopping condition. Herein, the stopping condition is the number of iterations (see Algorithm 4). However, in order to evaluate the performance of IRSPC two criteria have been considered: (i) the $\alpha\%$; percentage of the removed items, (ii) the total number of iterations.

Table 3 illustrates the performance of IRSPC when the number of iterations is fixed to 200 iterations, while, α is ranged according to the following, $\alpha = 10\%, 20\%, 30\%$, and 40%.

Table 3: The performance of the IRSPC with the variation of α

	$Variation\ of\ \alpha\ Iterations\ =\ 200$			
	$\alpha = 10\%$	$\alpha = 20\%$	$\alpha = 30\%$	$\alpha = 40\%$
Solution	338882	345898	342046	335164
Time (s)	12.316 s	14.83	20.18s	21.64

Table 3 shows the objective values and the solution times reached by the IRSPC. One can observed that, the best solution can be obtained with $\alpha = 20\%$. Moreover, the solution time is increase with the increasing of α. So, for the next step of the computational results, the α is fixed to 20%, while the number of iterations are ranged as follows: 100, 200, 300, and 400 iterations.

Table 4: The performance of the IRSPC with the variation of iterations

	$\alpha =\ 20\%\ \ Variation\ of\ Iterations$			
No. of iterations	100	200	300	400
Solution	345378	345912	346178	346432
Time (s)	7.8 s	14.21 s	21.37 s	31 s

Table 4 illustrates the performance of IRSPC when α is fixed to 20% and the number of iterations are varied. As it is clear that, the quality of solutions increased with the increasing of iterations, meanwhile the required solution times are increased. For 400 iterations, the algorithm yields best solutions within 31 seconds. In fact, the quality of solutions has priority, therefore the algorithm was tuned to 400 iterations for the next step of the computational results.

Table 5 shows the performance of IRSPC with compare with the greedy algorithm (Algorithm 1), and the local search procedure (Algorithm 2).

Table 5: The performance of IRSPC with compare with greedy and the local search

	Greedy	local search procedure	IRSPC
Solution	226362	271224	346432
Time (s)	0.015 s	0.047s	31 s

From Table 5, one can observed that, the performance of IRSPC for solving the considered problem is better than the both: greedy (Algorithm 1) and the local search (Algorithm 2). IRSPC produces high quality solution of 346432, while the greedy and the local search produce 226362 and 271224 respectively. Although, the required solution time for IRSPC is much more than those needed by the both algorithms. The IRSPC required about 31 seconds to produce its output, while the other algorithms, yield their outputs in 0.015, and 0.047 seconds respectively.

5. CONCLUSIONS

In this paper, a heuristic approach is proposed for solving a real-life situation with precedence constraints in agricultural land investment problem. The contribution in this work are that: first, the tackled problem has been simulated as a combinatorial optimization problem known as PCKP. Second, a mathematical representation model was proposed to represent the problem. Third and last, an iterative three-stage neighborhood search heuristic is proposed for solving the considered problem. The proposed solution method consists of three stages. First stage yields a feasible solution by using a greedy procedure. The greedy algorithm yields a fast solution of moderate quality. Second stage improves the solution at hand by using a local search method. The local search solution procedure improves the solution at hand but, it falls in a local optimum solution. Third and last stage, diversifies the solution search space by using a random neighborhood search technique. This technique proved its efficiency in escaping from a series of local optimum solutions. The three stages were iterated searching for the best local solution. The computational results show the effectiveness of the proposed heuristic algorithm for producing solutions of high quality in acceptable running time.

References

1. Aarts , E., & Lenstra, J. (2018). *Local search in combinatorial optimization.* Princeton University Press.
2. Aslan, A., Ursavas , E., & Romeijnders, W. (2023). A Precedence Constrained Knapsack Problem with Uncertain Item Weights for Personalized Learning Systems. *Omega*, 115, 102779.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication: Online Open Access
Vol: 42 Issue: 05-2023
DOI 10.17605/OSF.IO/KN46A

3. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., & Protasi, M. (2012). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Berlin Heidelberg: Springer Science & Business Media.
4. Boland, N., Bley, A., Fricke, C., Froyland, G., & Sotirov, R. (2012). Clique-based facets for the precedence constrained knapsack problem. *Mathematical Programming volume 133*, 481–511 .
5. Hifi, M., Saleh, S., & Wu, L. (2014). A Fast Large Neighborhood Search for Disjunctively Constrained Knapsack Problems. *Combinatorial Optimization volume 8596* (pp. 396-407). Springer, Cham.
6. Hifi, M., Saleh, S., & Wu, L. (2015). A hybrid guided neighborhood search for the disjunctively constrained knapsack problem. *Cogent Engineering, 2:1*, 1-25.
7. Kellerer, H., Pferschy, U., & Pisinger, D. (2013). *Knapsack Problems.* Berlin Hedelberg: Springer Science & Business Media.
8. Maiti, N., Pathak, P., & Samanta, B. (2021). An efficient algorithm for the precedence constraint knapsack problem with reference to large-scale open-pit mining pushback design. *Mining Technology, 130:1*, 8-21.
9. Nancel-Penard , P., Morales, N., & Cornillier, F. (2022). A recursive time aggregation-disaggregation heuristic for the multidimensional and multiperiod precedence-constrained knapsack problem: An application to the open-pit mine block sequencing problem. *European Journal of Operational Research*, 303(3), 1088-1099.
10. Pacheco, P. (2011). *An Introduction to Parallel Programming.* USA: Morgan Kaufmann Publishers.
11. Saleh, S. A. (2018). A Parallel Heuristic Method for Optimizing a Real Life Problem (Agricultural Land Investment Problem). *Academic Journal of Nawroz University, 7(4),*, 168–172.
12. Samphaiboon, N., & Yamada, Y. (2000). Heuristic and Exact Algorithms for the Precedence-Constrained Knapsack Problem. *Journal of Optimization Theory and Applications volume 105*, 659–676.
13. You , B., & Yamada, T. (2007). A pegging approach to the precedence-constrained knapsack problem. *European Journal of Operational Research, Volume 183, Issue 2*, 618-632.